

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

ADVANCED CLUSTER SYSTEMS, INC.,)	
)	
Plaintiff,)	Redacted - Public Version
)	
v.)	C.A. No. 19-2032-MN-CJB
)	
NVIDIA CORPORATION, NVIDIA)	
SINGAPORE PTE. LTD., and NVIDIA)	
INTERNATIONAL, INC.,)	
)	
Defendants.)	

JOINT CLAIM CONSTRUCTION BRIEF

John W. Shaw (No. 3362)
Karen E. Keller (No. 4489)
Nathan R. Hoeschen (No. 6232)
Emily S. DiBenedetto (No. 6779)
SHAW KELLER LLP
I.M. Pei Building
1105 North Market Street, 12th Floor
Wilmington, DE 19801
(302) 298-0700
jshaw@shawkeller.com
kkeller@shawkeller.com
nhoeschen@shawkeller.com
edibenedetto@shawkeller.com
*Attorneys for Plaintiff Advanced Cluster
Systems, Inc.*

Brian A. Biggs (DE Bar No. 5591)
Stephanie E. O'Byrne (DE Bar No. 4446)
Erin E. Larson (DE Bar No. 6616)
DLA PIPER LLP (US)
1201 North Market Street, Suite 2100
Wilmington, DE 19801-1147
(302) 468-5700
brian.biggs@us.dlapiper.com
stephanie.obyrne@us.dlapiper.com
erin.larson@us.dlapiper.com
*Attorneys for Defendants NVIDIA
Corporation, NVIDIA Singapore Pte Ltd.
and NVIDIA International*

Dated: December 21, 2022

TABLE OF CONTENTS

	Page No.
I. AGREED-UPON CONSTRUCTIONS	1
II. DISPUTED CONSTRUCTIONS	1
A. “Kernel” (claims 1, 26, 29, 35)	1
1. Plaintiff’s Opening Position.....	1
2. Defendants’ Answering Position	3
3. Plaintiff’s Reply Position.....	6
4. Defendants’ Sur-Reply Position	7
B. “Node”/“Nodes” (“Node(s)”) (claims 1, 4, 7–8, 10, 18–22, 24–26, 29–31, 33, 35, 36–38)	8
1. Plaintiff’s Opening Position.....	8
2. Defendants’ Answering Position	10
3. Plaintiff’s Reply Position.....	12
4. Defendants’ Sur-Reply Position	12
C. “Single-Node Kernel” (claims 1, 26, 29, 35).....	13
1. Plaintiff’s Opening Position.....	13
2. Defendants’ Answering Position	14
3. Plaintiff’s Reply Position.....	16
4. Defendants’ Sur-Reply Position	17
D. “Third Node” Phrase (claims 1, 26, 29).....	17
1. Plaintiff’s Opening Position.....	18
2. Defendants’ Answering Position	20
3. Plaintiff’s Reply Position.....	24

TABLE OF CONTENTS
(*cont'd*)

	Page No.
a. NVIDIA fails to show clear and unequivocal disclaimer	24
b. NVIDIA’s “configured to” arguments do not support its construction	25
4. Defendants’ Sur-Reply Position	26
E. “Peer-To-Peer Architecture” (claims 1, 35).....	27
1. Plaintiff’s Opening Position.....	27
2. Defendants’ Answering Position	29
3. Plaintiff’s Reply Position.....	33
4. Defendants’ Sur-Reply Position	35
F. “Cluster Node Module” (claims 4, 7, 8, 10, 27, 30, 31)	37
1. Plaintiff’s Opening Position.....	37
2. Defendants’ Answering Position	38
3. Plaintiff’s Reply Position.....	40
4. Defendants’ Sur-Reply Position	41
G. “Mechanism” Terms (claims 1, 26, 29, 35)	42
1. Plaintiff’s Opening Position.....	43
a. Section 112, ¶ 6 does not apply.	43
b. If § 112, ¶ 6 applies, the Court should adopt ACS’s proposed functions.	47
c. If § 112, ¶ 6 applies, the Court should adopt ACS’s proposed structures.	48
2. Defendants’ Answering Position	49
a. The mechanism claim terms are subject to § 112, ¶ 6	49

TABLE OF CONTENTS
(*cont'd*)

	Page No.
b. The Court should adopt NVIDIA’s proposed functions.....	53
c. The Court should adopt NVIDIA’s proposed structures	54
3. Plaintiff’s Reply Position.....	57
a. NVIDIA fails to rebut the presumption that § 112, ¶ 6 does not apply	57
b. NVIDIA all but concedes its functions are incorrect.....	60
c. NVIDIA includes unnecessary components while ignoring equivalents to propose an unjustifiably narrow structure	60
4. Defendants’ Sur-Reply Position	63
H. “First Node” Term (claim 35).....	65
1. Plaintiff’s Opening Position.....	66
a. “The first node” is an obvious error.....	66
b. The “second node” correction is not subject to reasonable debate and is supported by the prosecution history.	66
2. Defendants’ Answering Position	68
3. Plaintiff’s Reply Position.....	70
4. Defendants’ Sur-Reply Position	72
I. “Configured to Access ... Comprising ...” Terms (claims 1, 26, 29, 35)	72
1. Plaintiff’s Opening Position.....	72
2. Defendants’ Answering Position	73

TABLE OF CONTENTS
(*cont'd*)

	Page No.
3. Plaintiff's Reply Position.....	74
4. Defendants' Sur-Reply Position	78

TABLE OF AUTHORITIES

Page No(s).

<i>Acuity Brands Lighting, Inc. v. Ultravision Techs, LLC</i> 19-cv-2207-MN, 2021 WL 3187439 (D. Del., Jul. 28, 2021)	<i>passim</i>
<i>Advanced Fiber Tech. Tr. v. J & L Fiber Svcs., Inc.,</i> 674 F.3d 1365 (Fed. Cir. 2012).....	10, 11
<i>Alarm.com, Inc. v. SecureNet Techs., LLC,</i> 2019 WL 3996883 (D. Del. July 23, 2019)	65
<i>Aspex Eyewear, Inc. v. Marchon Eyewear, Inc.,</i> 672 F.3d 1335 (Fed. Cir. 2012).....	<i>passim</i>
<i>Asyst Techs., Inc. v. Empak, Inc.,</i> 268 F.3d 1364 (Fed. Cir. 2001).....	60, 62
<i>Aylus Networks, Inc. v. Apple, Inc.,</i> 856 F.3d 1353 (Fed. Cir. 2017).....	23, 26, 37
<i>Ball Aerosol & Specialty Container, Inc. v. Ltd. Brands, Inc.,</i> 555 F.3d 984 (Fed. Cir. 2009).....	76
<i>Bayer Intell. Prop. GmbH v. Taro Pharm. Indus. Ltd.,</i> 2019 WL 1466193 (D. Del. Apr. 2, 2019).....	66
<i>BBA Nonwovens Simpsonville, Inc. v. Superior Nonwovens, LLC,</i> 303 F.3d 1332 (Fed. Cir. 2002).....	47, 60
<i>Bicon, Inc. v. Straumann Co.,</i> 441 F.3d 945 (Fed. Cir. 2006).....	7
<i>Carrum Techs., LLC v. Unified Patents, LLC,</i> 2020-2204, 2021 WL 3574209 (Fed. Cir. Aug. 13, 2021)	7
<i>Cupp Comut. AS v. Trend Micro Inc.,</i> 53 F.4th 1376 (Fed. Cir. 2022)	26, 37
<i>Cutsforth, Inc. v. Westinghouse Air Brake Techs. Corp.,</i> No. 17-1025, 2018 WL 4076837 (W.D. Pa. Aug. 27, 2018).....	18, 19
<i>Dyfan, LLC v. Target Corp.,</i> 28 F.4th 1360 (Fed. Cir. 2022)	57, 58, 59, 63

TABLE OF AUTHORITIES

Page No(s).

<i>Egenera, Inc. v. Cisco Sys., Inc.</i> , 972 F.3d 1367 (Fed. Cir. 2020).....	63
<i>Ethicon LLC v. Intuitive Surgical, Inc.</i> , 847 Fed. App'x. 901 (Fed. Cir. 2021).....	30
<i>Exergen Corp. v. Wal-Mart Stores, Inc.</i> , 575 F.3d 1312 (Fed. Cir. 2009).....	75
<i>Extang Corp. Undercover, Inc. v. Truck Accessories Grp., LLC</i> , 2020 WL 6888277 (D. Del. Nov. 24, 2020)	46, 50
<i>Genentech, Inc. v. Chiron Corp.</i> , 112 F.3d 495 (Fed. Cir. 1997).....	74, 77
<i>Generation II Orthotics, Inc. v. Med Tech., Inc.</i> , 263 F.3d 1356 (Fed. Cir. 2001).....	47, 60
<i>Grecia v. Samsung Elecs. Am., Inc.</i> , 780 Fed. App'x. 912 (Fed. Cir. 2019).....	<i>passim</i>
<i>Group One v. Hallmark Cards</i> , 407 F.3d 1297 (Fed. Cir. 2005).....	69, 71
<i>Hewlett-Packard Co. v. Bausch & Lomb Inc.</i> , 909 F.2d 1464 (Fed. Cir. 1990).....	19
<i>Hoffer v. Microsoft Corp.</i> , 405 F.3d 1326 (Fed. Cir. 2005).....	71, 72
<i>IGT v. Bally Gaming Int'l, Inc.</i> , 659 F.3d 1109 (Fed. Cir. 2011).....	16, 17
<i>Indacon, Inc. v. Facebook, Inc.</i> , 824 F.3d 1352 (Fed. Cir. 2016).....	14, 15, 16
<i>Intell. Ventures I LLC v. AT & T Mobility LLC</i> , 2015 WL 1393386 (D. Del. Mar. 24, 2015)	38
<i>Interval Licensing, LLC v. AOL, Inc.</i> , 766 F.3d 1364 (Fed. Cir. 2014).....	37

TABLE OF AUTHORITIES

Page No(s).

<i>INVT SPE LLC v. Int’l Trade Comm’n</i> , 46 F.4th 1361 (Fed. Cir. 2022)	<i>passim</i>
<i>Irdeto Access, Inc. v. Echostar Satellite Corp.</i> , 383 F.3d 1295 (Fed. Cir. 2004).....	39
<i>JVW Enters., Inc. v. Interact Accessories, Inc.</i> , 424 F.3d 1324 (Fed. Cir. 2005).....	48
<i>Kyocera Senco Indus. Tools Inc. v. Int’l Trade Comm’n</i> , 22 F.4th 1369 (Fed. Cir. 2022)	<i>passim</i>
<i>Linear Tech. Corp. v. Int’l Trade Comm’n</i> , 566 F.3d 1049 (Fed. Cir. 2009).....	10
<i>Mad Dogg Athletics, Inc. v. Peloton Interactive, Inc.</i> , 2021 WL 3200994 (E.D. Tex. July 28, 2021)	58
<i>Mars, Inc. v. H.J. Heinz Co., L.P.</i> , 377 F.3d 1369 (Fed. Cir. 2004).....	74, 75, 77
<i>Mass. Inst. of Tech. v. Abacus Software</i> , 462 F.3d 1344 (Fed. Cir. 2006).....	49
<i>Mass. Inst. of Tech. v. Shire Pharms., Inc.</i> , 839 F.3d 1111 (Fed. Cir. 2016).....	18, 19
<i>Med. Instrumentation & Diagnostics Corp. v. Elekta AB</i> , 344 F.3d 1205 (Fed. Cir. 2003).....	56, 62
<i>Media Rights Tech., Inc. v. Cap. One Fin. Corp.</i> , 800 F.3d 1366 (Fed. Cir. 2015).....	50, 58
<i>Microsoft Corp. v. Int’l Trade Comm’n</i> , 731 F.3d 1354 (Fed. Cir. 2013).....	10
<i>Midwest Athletics and Sports Alliance LLC v. Xerox Corp.</i> , 2020 WL 7692767 (W.D.N.Y. Dec. 28, 2020).....	46, 50
<i>MTD Prods. Inc. v. Iancu</i> , 933 F.3d 1336 (Fed. Cir. 2019).....	<i>passim</i>

TABLE OF AUTHORITIES

Page No(s).

<i>Nautilus, Inc. v. Biosig Instruments, Inc.</i> , 572 U.S. 898 (2014).....	68
<i>Nazomi Comm'ns, Inc. v. Nokia Corp.</i> , 739 F.3d 1339 (Fed. Cir. 2014).....	79
<i>Nevro Corp. v. Boston Sci. Corp.</i> , 955 F.3d 35 (Fed. Cir. 2020).....	79
<i>Northrop Grumman Corp. v. Intel Corp.</i> , 325 F.3d 1346 (Fed. Cir. 2003).....	49, 60
<i>Novartis Pharms. Corp. v. Accord Healthcare, Inc.</i> , 38 F.4th 1013 (Fed. Cir. 2022)	34
<i>Novo Indus., L.P. v. Micro Molds Corp.</i> , 350 F.3d 1348 (Fed. Cir. 2003).....	69
<i>Omega Eng'g, Inc. v. Raytek Corp.</i> , 334 F.3d 1314 (Fed. Cir. 2003).....	30
<i>On-Line Techs. Inc. v. Bodenseewerk Perkin-Elmer GMBH</i> , 386 F.3d 1133 (Fed. Cir. 2004).....	31
<i>Parkervision, Inc. v. LG Elecs., Inc.</i> , 2022 WL 2240465 (W.D. Tex. June 21, 2022)	12
<i>ParkerVision, Inc. v. Qualcomm Inc.</i> , 903 F.3d 1354 (Fed. Cir. 2018).....	20, 76
<i>Personalized Media Commc'ns, LLC v. Int'l Trade Comm'n</i> , 161 F.3d 696 (Fed. Cir. 1998).....	44, 45, 50, 58
<i>Radware, Ltd. v. A10 Networks, Inc.</i> , 13-cv-2024, 2014 WL 1572644 (N.D. Cal. Apr. 18, 2014).....	21, 77
<i>Signtech USA, Ltd. v. Vutek, Inc.</i> , 174 F.3d 1352 (Fed. Cir. 1999).....	65
<i>Smith & Nephew, Inc. v. Arthrex, Inc.</i> , 355 Fed. App'x 384 (Fed. Cir. 2009).....	5

TABLE OF AUTHORITIES

Page No(s).

<i>SuperGuide Corp. v. DirecTV Enters., Inc.</i> , 358 F.3d 870 (Fed. Cir. 2004).....	2, 3
<i>Synchronous Techs., Inc. v. Dropbox, Inc.</i> , 987 F.3d 1358 (Fed. Cir. 2021).....	56, 62
<i>Thorner v. Sony Comput. Ent. Am. LLC</i> , 669 F.3d 1362 (Fed. Cir. 2012).....	<i>passim</i>
<i>TQ Delta LLC v. Adtran, Inc.</i> , 14-cv-954-RGA, 2021 WL 1200595 (D. Del. Mar. 30, 2021)	79
<i>Triplay, Inc. v. Whatsapp, Inc.</i> , 2016 WL 3574012 (D. Del. June 30, 2016).....	58, 64
<i>Triton Tech of Texas, LLC v. Nintendo of Am., Inc.</i> , 753 F.3d 1375 (Fed. Cir. 2014).....	56, 62, 63
<i>Trustees of Columbia Univ. v. Symantec Corp.</i> , 811 F.3d 1359 (Fed. Cir. 2016).....	5
<i>Typhoon Touch Techs., Inc. v. Dell, Inc.</i> , 659 F.3d 1376 (Fed. Cir. 2011).....	20, 76
<i>UltimatePointer, L.L.C. v. Nintendo Co.</i> , 816 F.3d 816 (Fed. Cir. 2016).....	19
<i>Unicorn Glob. Inc. v. Golabs, Inc.</i> , 2020 WL 2745692 (N.D. Tex. May 26, 2020)	47, 50
<i>VirnetX Inc. v. Apple Inc.</i> , 792 Fed. App'x 796 (2019).....	4
<i>Vistan Corp. v. Fadei USA, Inc.</i> , 547 F. App'x 986 (Fed. Cir. 2013) (unpublished)	62, 64
<i>Vitronics Corp. v. Conceptronic, Inc.</i> , 90 F.3d 1576 (Fed. Cir. 1996).....	1, 4, 16, 32
<i>Welker Bearing Co. v. PHD, Inc.</i> , 550 F.3d 1090 (Fed. Cir. 2008).....	49, 57

TABLE OF AUTHORITIES

Page No(s).

<i>Williamson v. Citrix Online, LLC</i> , 792 F.3d 1339 (Fed. Cir. 2015).....	49, 51, 57, 58
<i>Zeroclick, LLC v. Apple Inc.</i> , 891 F.3d 1003 (Fed. Cir. 2018).....	43, 44, 50, 58

OTHER AUTHORITIES

35 U.S.C. § 112.....	<i>passim</i>
LR 7.1.3	37, 64
Dictionary of IEEE Standards Terms, 804 (7th ed. 2000).....	32

TABLE OF EXHIBITS

Exhibit	Description	Short-Hand
A	Amended Joint Claim Construction Chart (D.I. 254)	
B	U.S. Patent No. 10,333,768	'768
C	Prosecution History of U.S. Patent No. 10,333,768	
D	Expert Report of Jaswinder Pal Singh, Ph. D. On Claim Construction	
E	Declaration Of Dr. Ian Foster	Foster Decl.
F	Declaration Of Henry Tufo, IPR2021-00019, Ex. 1005	
G	Declaration Of Henry Tufo, IPR2021-00020, Ex. 1105	
H	Defendants' Final Invalidity Contentions	
I	Defendants' 3rd Supplemental Responses & Objections to Plaintiff's 6th Set of Interrogatories (No. 10) & 2nd Supplemental Responses and Objections To Plaintiff's 7th Set Of Interrogatories (Nos. 11–20)	
J	NVLink 2.0 Specification, Rev. 1.03 (NVIDIA-ACS-0146862)	
K	NVLink 2.0 Specification, Rev. 0.575 (NVIDIA-ACS-0089434)	
L	NVIDIA Coherent Interconnect (NCI) Specification (NVIDIA-ACS-0086665)	
M	S. Wolfram, <i>The Mathematica Book</i> (5th Ed. 2003)	
N	Patent Owner Preliminary Response, IPR2021-00019	
O	Patent Owner Preliminary Response, IPR2021-00020	
P	Decision Denying Institution of <i>Inter Partes</i> Review, IPR2021-00019	
Q	Decision Denying Institution of <i>Inter Partes</i> Review, IPR2021-00020	
R	NVIDIA Final Infringement Contentions, Ex. A23 (Polymath System)	
S	<i>Adidas AG v. Under Armour, Inc.</i> , 14-cv-130, Doc. 122 (D. Del. Dec. 15, 2015)	
T	Petition For <i>Inter Partes</i> Review of U.S. Patent No. 10,333,768, IPR2021-00019	
U	Petition For <i>Inter Partes</i> Review of U.S. Patent No. 10,333,768, IPR2021-00019	
V	<i>Parkervision, Inc. v. Hisense Co.</i> , No. 6-20-cv-00870-ADA, Dkt. 51 (W.D. Tex. Aug. 29, 2022)	
W	U.S. Patent No. 8,082,289	
1	U.S. Patent No. 10,333,768, April 13, 2018, Office Action	
2	Decision Granting Institution of <i>Inter Partes</i> Review, IPR2020-01608	
3	Decision Granting Institution of <i>Inter Partes</i> Review, IPR2020-00075	
4	Decision Granting Institution of <i>Inter Partes</i> Review, IPR2020-00108	
5	2008 ACS SEM Presentation	
6	IPR2021-00020 - Ex. 2001 - Singh Declaration	
7	IPR2021-00019 - Ex. 2001 - Singh Declaration	

I. Agreed-Upon Constructions

The parties do not agree on any constructions.

II. Disputed Constructions**A. “Kernel” (claims 1, 26, 29, 35)**

ACS	Plain and ordinary meaning: program code
NVIDIA	program code for interpreting high-level code, commands, and/or instructions supplied by a user or a script into low-level code, such as, for example, machine language or assembly language

1. Plaintiff’s Opening Position

NVIDIA’s proposed construction is not the ordinary meaning for the term “kernel,” and is unsupported by the intrinsic record. “Kernel” was a well-known term to a POSITA,¹ Ex. D, ¶ 78, who would have understood “kernel” to mean program code. *See* Ex. B (’768 Patent), Cls. 1, 12, 16, 26, 29, 35 (referring to a “kernel” as “program code”), 1:29–33; Ex. D, ¶¶ 78–80. NVIDIA replaces this easily understood term with a needlessly complex and limiting construction.

The claims themselves best describe the varied configurations of the “kernel.” Ex. B, Cls. 1, 26, 29, 30, 35 (evaluating mathematical expressions), Cls. 1, 26, 29, 35 (interpreting user instructions and distributing calls), Cls. 5, 25 (accepting and executing requests), Cls. 27, 28, 32 (calling commands and sending packets), Cl. 32 (being associated with a cluster node module, and receiving expressions and updating variables), Cl. 2 (comprising Mathematica); Ex. D, ¶¶ 78–80. No further construction is needed. *Vitronics Corp. v. Conceptronic, Inc.*, 90 F.3d 1576, 1582 (Fed. Cir. 1996) (“the claims themselves ... define ... the patented invention”).

¹ All references to a POSITA are to a person of ordinary skill in the art at the time of the invention in the field of cluster computing. Ex. D, ¶¶ 34–36.

NVIDIA's construction (1) requires unclaimed functionality (*e.g.*, interpreting high-level code ... into low-level code) improperly narrowing the claims, and (2) duplicates claimed functionality (*e.g.*, interpreting user instructions) rendering some claim language surplusage. Ex. D, ¶¶ 81–82; Ex. B, Cls. 1, 26, 29, 35. NVIDIA's construction attempts to improperly import the limitation, “for interpreting high-level code, commands, and/or instructions supplied by a user or a script into low-level code, such as, for example, machine language or assembly language,” from the specification into the claims. Ex. D, ¶¶ 81–82; *SuperGuide Corp. v. DirecTV Enters., Inc.*, 358 F.3d 870, 875 (Fed. Cir. 2004) (“a particular embodiment ... may not be read into a claim”).

Generally, claim terms are given their plain and ordinary meaning except: “1) when a patentee sets out a definition and acts as his own lexicographer, or 2) when the patentee disavows the full scope of a claim term either in the specification or during prosecution.” *Thorner v. Sony Comput. Ent. Am. LLC*, 669 F.3d 1362, 1365–67 (Fed. Cir. 2012). Neither exception applies.

The specification does not “clearly set forth a definition of [kernel] other than its plain and ordinary meaning.” *Id.* at 1365. FIGS. 1 and 2 show “**one embodiment** of a cluster system 100” that includes “kernel module[] 206.” Ex. B, 11:15–18. “[K]ernel module 206 typically includes **program code** for interpreting high-level code” *Id.*, 23:8–15. NVIDIA's construction is based on a portion of the specification that discusses a preferred embodiment; this portion is neither definitional nor a disavowal. *Id.*, 23:8–26; Ex. D, ¶¶ 84–85. Kernel module 206 relates to a “typical” configuration of “one embodiment.” Ex. B, 4:1–3, 11:15–18; Ex. D, ¶ 85. This portion of the specification also does not purport to comprehensively describe the kernel module: it is limited to one feature that is typically “include[d].” Ex. B, 23:9. For the same reasons, this is not a “clear disavowal” of claim scope, such as an “expression[] of manifest exclusion or restriction.”

Thorner, 669 F.3d at 1365; Ex. B, 23:8–26, 1:29–37; Ex. D, ¶¶ 84–87. NVIDIA’s attempt to import limitations from the specification is improper. *Thorner*, 358 F.3d at 1366.

2. Defendants’ Answering Position

The claim term “kernel” is not synonymous with “program code,” as ACS contends. Whereas “program code” refers to application software generally, “kernel” refers in the claims to a specific feature of certain application software.

The ’768 patent purports to add “computer cluster” functionality to application programs that do not already have that capability. The ’768 patent asserts that “[m]any computer application programs are not currently designed to benefit from advantages that computer clusters can offer.” ’768 at 1:22–26. The patent then points to a feature in certain programs that can be used to expand their functionality: “Some application programs include an interpreter that executes instructions provided to the program by a user, a script, or another source.” *Id.* at 1:29–31. The patent refers to this interpreter feature as a “kernel.” *Id.*, at 1:31–37 (“Such an interpreter is sometimes called a ‘kernel’....”); Foster Decl., ¶ 52; *Kyocera Senco Indus. Tools Inc. v. Int’l Trade Comm’n*, 22 F.4th 1369, 1378–79 (Fed. Cir. 2022) (construing term “driven position” based, in part, on statement in specification that “[t]his bottom position is also sometimes referred to herein as the ‘driven position’”). The claimed “kernel” converts “high-level” inputs into “low-level” outputs for the computer hardware. ’768 at 23:9–12. In fact, NVIDIA’s construction directly quotes the specification language describing the claimed “kernel,” namely: “program code for interpreting high-level code, commands, and/or instructions supplied by a user or a script into low-level code, such as, for example, machine language or assembly language.” *Id.*

ACS mistakenly argues that a POSITA would have understood “kernel” to be coextensive with “program code.” ACS first offers a conclusory statement from its expert (Joint Br. at 1 (citing

Ex. D, ¶¶ 78–80)) but conclusory expert opinions are entitled to no weight, particularly where, as here, they contradict the intrinsic record. *Vitronics Corp. v. Conceptronic, Inc.*, 90 F.3d 1576, 1584 (Fed. Cir. 1996). ACS also argues that the claim itself “refer[s] to a ‘kernel’ as ‘program code.’” Joint Br. at 1. To the contrary, the claim does not equate the meaning of “kernel” with that of “program code,” but instead refers to the claimed kernel as being implemented in program code, *e.g.*, “program code for a single-node kernel.” Indeed, ACS’ proposed construction would render the claim language nonsensical, improperly rewriting it as “program code for a single-node [program code].” *Id.* at 4.

ACS also argues that “NVIDIA’s construction (1) requires unclaimed functionality ... and (2) duplicates claimed functionality (*e.g.*, interpreting user instructions) rendering some claim language surplusage.” Joint Br. at 2. ACS’ argument (1) relies on its assertion that “kernel” means “program code,” which fails for the reasons above. And contrary to ACS’ argument, (2) none of the additional claim limitations are rendered “surplusage” by NVIDIA’s construction. For example, NVIDIA’s construction includes several different types of inputs from two different sources, *i.e.*, “high-level code, commands, and/or instructions supplied by a user or a script.” The limitation raised by ACS (“interpreting user instructions”) (Joint Br. at 1) further narrows the scope of that claimed kernel, requiring it to be “configured to interpret user instructions” specifically, only one of the options in NVIDIA’s construction of “kernel.” *See* ’768 at claim 1. Thus, nothing in NVIDIA’s construction would render that limitation surplusage. *See VirnetX Inc. v. Apple Inc.*, 792 Fed. App’x 796, 810–11 (2019). Nor does anything in the specific “configurations of the ‘kernel’” recited in other parts of the claim (Joint Br. at 1) justify reading out the claim requirement that a “kernel” specifically—not just any program code generally—perform the recited functions.

ACS further argues that its broader construction must be adopted unless NVIDIA can show lexicography or disclaimer. Joint Br. at 2 (citing *Thorner v. Sony Comput. Ent. of Am.*, 669 F.3d 1362, 1365–67 (Fed. Cir. 2012)). Lexicography or disclaimer is not required where, as here, NVIDIA’s construction relies on the plain and ordinary meaning of the term in the context of the specification. *Trustees of Columbia Univ. v. Symantec Corp.*, 811 F.3d 1359, 1364 (Fed. Cir. 2016) (*Thorner* does not support the “argument that the presumption of plain and ordinary meaning ‘can be overcome in only two circumstances: [when] the patentee has expressly defined a term or has expressly disavowed the full scope of the claim in the specification and the prosecution history,’” and affirming that “[t]he only meaning that matters in claim construction is the meaning in the context of the patent.”); Foster Decl., ¶¶ 34–38, 46–55 (discussing the context-specific meaning of “kernel” in the ’768 patent).

Nor does the specification’s use of words like “embodiment” or “typically” require ACS’ construction. The specification states that “[a] kernel module typically includes program code for interpreting high-level code, commands, and/or instructions supplied by a user or a script into low-level code, such as, for example, machine language or assembly language.” ’768 at 23:9–12. This passage states only that a kernel would typically be able to take any of those inputs and provide any of those outputs. The claimed kernel, however, may be narrower than this, such as the claimed kernel discussed above that must be configured to use one type of input (*i.e.*, “instructions”) from one type of source (*i.e.*, a “user”). In any event, the claim scope cannot be construed broader the ordinary meaning of “kernel,” as described in the specification, regardless of whether the specification uses words like “embodiment” or “typically.” See *Smith & Nephew, Inc. v. Arthrex, Inc.*, 355 Fed. App’x 384, 386–87 (Fed. Cir. 2009). Accordingly, the Court should reject ACS’ attempt to define the term “kernel” so broadly as to essentially read it out of the claims.

3. Plaintiff's Reply Position

NVIDIA relies on the specification's explanation that some programs include an interpreter, and later reference to this interpreter as a "kernel" to conclude that the kernel is an interpreter that functions in precisely the manner described in a preferred embodiment. Despite its claims, NVIDIA's analysis is rooted in lexicography as confirmed by NVIDIA's cases. *Kyocera*, 22 F.4th at 1378 (finding lexicography defined the claim term); *Columbia Univ.*, 811 F.3d at 1363 (same). NVIDIA's arguments further demonstrate that NVIDIA plucks its definition from the specification. Joint Br. at 3 (arguing NVIDIA's construction "directly quotes the specification language"), 5 (arguing "ordinary meaning of 'kernel,' as *described in the specification*.").

NVIDIA cannot show the *clear intent* to redefine a term required to establish its lexicography argument. NVIDIA relies heavily on the passage, "[s]uch an interpreter is sometimes called a 'kernel,'" to supply its definition. But this is the opposite of a definition because "sometimes" confirms that interpreters are not always called kernels. *Kyocera* does not suggest otherwise because that specification stated that "[t]his [] position is also sometimes *referred to herein* as the 'driven position,'" 22 F.4th at 1379. The '768 Patent never states the kernel "herein" refers to an interpreter. The specification also expressly differentiates interpreters from kernels by later referring to "an interpreter *or* a kernel." Ex. B, 4:14–33. Even if the specification could be read to say an interpreter is a kernel (which it does not), it does not follow that a kernel is an interpreter. A tractor is "sometimes called" a vehicle, but not all vehicles are tractors. Likewise a kernel may include an interpreter, but not all kernels are interpreters.

NVIDIA misrepresents ACS's construction. ACS proposes plain and ordinary meaning, not program code. ACS's articulation of "kernel" as "program code" was included "[t]o clarify

the disputes between the parties.” D.I. 247. It does so by highlighting that both parties agree that “kernel” is program code, but disagree whether that program code must be for interpreting high-level code into low-level code. As discussed above, absent lexicography or disclaimer, this additional limitation is improper. The specification uses “kernel” to identify the computational portion of a computer program, as opposed to the front end or user interface portion. *See* Ex. B, 2:30–3:26. ACS’s construction should be adopted because it articulates the ordinary meaning of kernel *as used in the claims*. It does not capture all possible nuances of a kernel as it would have been understood by a POSITA based on the specification. It does not need to, because the relevant characteristics of the claimed “kernel” are defined by the claims. Ex. D, ¶¶ 79–80. Incorporating explicitly claimed functionality into the articulation of the plain and ordinary meaning of “kernel” would be redundant.

4. Defendants’ Sur-Reply Position

ACS seeks to strip the claim term “kernel” of any independent meaning, arguing that its meaning is “defined by” other limitations. Joint Br. at 7. Ignoring the intrinsic meaning of “kernel” is “contrary to the principle that claim language should not be treated as meaningless.” *Bicon, Inc. v. Straumann Co.*, 441 F.3d 945, 951 (Fed. Cir. 2006). Nor is NVIDIA attempting to, or required to, redefine the term “kernel” via lexicography, as ACS mistakenly argues. Joint Br. at 6. Instead, “kernel” itself has a plain meaning in the context of the patent, referring to an interpreter that converts “high-level” instructions into “low-level” code; this also is the accepted meaning in the field of mathematical evaluation software which the patent claims to improve. ’768 at 23:9–12, 1:29–37, 30:31–43, Abstract; Foster Decl. ¶¶ 34–38 (extrinsic uses of “kernel” in the art); *id.*, ¶¶ 39–63 (intrinsic evidence). *Carrum Techs., LLC v. Unified Patents, LLC*, 2020-2204, 2021 WL 3574209, at *5–6 (Fed. Cir. Aug. 13, 2021). In this regard, contrary to ACS, the use of

the disjunctive “or” in the specification’s reference to “an interpreter or a kernel” is consistent with the terms being used interchangeably in the patent. ’768 at 4:13–23; Joint Br. at 6.

ACS also asserts—in a single sentence—that “[t]he specification uses ‘kernel’ to identify the computational portion of a computer program.” Joint Br. at 7. If ACS is proposing a new construction, it is untimely and waived. In any event, the specification passage cited by ACS does not use the phrase “computational portion of a computer program” but instead confirms NVIDIA’s construction by describing, *e.g.*, a “first kernel ... configured to translate commands into code for execution on a first processor.” ’768 at 2:34–37.

B. “Node”/“Nodes” (“Node(s)”) (claims 1, 4, 7-8, 10, 18-22, 24-26, 29-31, 33, 35, 36-38)

ACS	Plain and ordinary meaning: computing device[s] (<i>e.g.</i> , computer[s], microprocessor[s], special purpose microprocessor[s], and/or processor core[s]) that can intercommunicate with other nodes
NVIDIA	a processing unit or subunit that is capable of single-threaded execution of code

1. Plaintiff’s Opening Position

A POSITA would have understood “node(s)” according to its ordinary meaning in view of the ’768 Patent: computing device[s] (*e.g.*, computer[s], microprocessor[s], special purpose microprocessor[s], and/or processor core[s]) that can intercommunicate with other nodes. Ex. B, 1:19–22, 8:39–40. NVIDIA replaces this simple construction with an improperly narrow one.

A POSITA would have understood “node(s)” to simply be computing devices, which can intercommunicate with other nodes. The claims emphasize the breadth of node hardware, including single-core, multicore, and special purpose microprocessors. *Id.*, Cls. 1 (“a first node comprising a ... processor ... a second node comprising a ... processor with a plurality of ... cores”), 26 (same), 29 (same); Cl. 35 (substantially similar), Cl. 39 (“the ... processor comprises a special purpose microprocessor”). The claims require that the nodes be able to intercommunicate

with other nodes. *Id.*, Cls. 1, 26, 29, 35. The specification specifies that these computing devices exemplarily include computers, microprocessors, special purpose microprocessors, and processing cores as long as those devices can intercommunicate. *Id.*, 1:19–21 (“computers, microprocessors, and/or processor cores (‘nodes’) that intercommunicate”), 4:42–45, 8:39–40 (“processors ... can be any special purpose microprocessor such as a digital signal processor”), 9:57–59, 7:21–23. The prosecution history does not contradict this disclosure.

NVIDIA’s construction improperly imports limitations from the specification. Lexicography does not apply here. The specification contains no clear expression of intent to define node(s). *Thorner*, 669 F.3d at 1365. The specification’s broad use of node(s) indicates an intent against defining the term narrowly. Ex. B, 1:19–20, 4:42–45, 7:21–23, 8:39–40, 9:57–59. Indeed, the sentence before the portion of the specification NVIDIA seeks to import describes “one embodiment” that “for example” “can include.” *Id.*, 4:42–44. Moreover, the phrase “capable of single-threaded execution of code” is irrelevant to whether the node can function in a cluster.

Further, NVIDIA’s own arguments and internal documents belie NVIDIA’s improperly narrow construction. Neither NVIDIA’s IPR expert over five IPRs, *see, e.g.*, Ex. F at 24; Ex. G at 24–25, nor NVIDIA’s current expert, as recently as October 2021, construed “node(s),” *see* Ex. E, § VI (Opinions Presented). Indeed, NVIDIA’s expert concedes the breadth of node hardware. Ex. E, ¶ 25 (“node can be a single *microprocessor or a computer*”); *see also id.*, ¶¶ 90, 99. Moreover, some NVIDIA arguments rely on a computer system as a node. *See, e.g.*, Ex. H at 70 (“Each node [of the cluster] consists of a *dual-CPU HP PC*”), 109–10, 132 (“devices” are “nodes”). Other NVIDIA arguments rely on a CPU or GPU as nodes. *See, e.g., id.* at 113 (“[a] program intended to be run on both a CPU and one or more GPUs is a multi-node program”); *see also id.* at 109–10; Ex. I at 134. Finally, even NVIDIA’s internal technical specifications refer to GPUs as nodes.

See, e.g., Ex. J at -884 (“An NVLink system is comprised of two or more ... ‘nodes’. Legal NVLink node[] [sic] types are:” “CPU” and “GPU”); Ex. K at -696; Ex. L at -681.

2. Defendants’ Answering Position

The patent specification expressly defines “node”: “The term ‘node’ refers to a processing unit or subunit that is capable of single-threaded execution of code.” ’768 at 4:45–47. This lexicography controls. *Advanced Fiber Tech. Tr. v. J & L Fiber Svcs., Inc.*, 674 F.3d 1365, 1372 (Fed. Cir. 2012) (“[I]f the specification ... defines a claim term, that definition shall apply even if it differs from the term’s ordinary meaning.”); *Microsoft Corp. v. Int’l Trade Comm’n*, 731 F.3d 1354, 1360 (Fed. Cir. 2013) (“The specification defines what is meant by ‘state’: ‘The ‘state’ of the contents of a collection stored at a server *refers to* the identity of the current version of a resource stored at a server.”) (emphasis added); *Linear Tech. Corp. v. Int’l Trade Comm’n*, 566 F.3d 1049, 1054 (Fed. Cir. 2009) (finding a “definition in the [patent-in-suit’s] specification,” where it “reads ‘[a]s used herein, the term ‘synchronously-switched switch’ *refers to* a switch including....’”) (emphasis added).

Each of ACS’ arguments to the contrary lacks merit. First, ACS posits that a POSITA would have understood “node” to mean ACS’ construction, yet ACS’ expert, Dr. Singh, is silent on the subject in his declaration. Joint Br. at 8-9. By contrast, NVIDIA’s expert, Dr. Foster, confirms the express teaching of the specification is correct. Foster Decl., ¶ 25.

Second, the claims do not conflict with the specification’s lexicography. The claim language cited by ACS simply states that the “node” “comprises” (*i.e.*, includes) a “processor” or a “microprocessor.” Joint Br. at 8–9. That is consistent with the specification’s lexicography stating that a “node” “refers to a processing unit or subunit.” ’768 at 4:45–47.

Third, other parts of the specification do not contradict the lexicography. Joint Br. at 8–9. Indeed, to the contrary, the specification consistently teaches single-threaded execution by processing units in every embodiment. ’768 at 5:6–20, 5:61–64, 7:23–31; 8:41–49, 9:59–67, 11:30–32. By contrast, the first specification passage cited by ACS (*id.* at 1:19–22) simply notes that a node may take the form of a computer, microprocessor, or processor core. This statement is entirely consistent with the lexicography that a “node” “refers to a processing unit or subunit” and does not contradict the lexicography’s statement (and the rest of the specification’s consistent teaching) that the claimed node must be “capable of single-threaded execution of code.” The second sentence cited by ACS (*id.* at 4:42–45) simply states, again, that a node can take the form of a processor core, and that sentence is literally followed by the specification’s lexicography of “node.” None of the other sentences ACS cites even uses the word “node.” *Id.* at 7:21–23, 8:39–40, 9:57–59.

Fourth, the fact that neither party previously construed “node” cannot somehow erase the patentee’s lexicography, and ACS cites no authority that it can. Joint Br. at 9.

Fifth, nothing in Dr. Foster’s declaration or prior NVIDIA arguments is contrary to the lexicography and could not override the lexicography in the intrinsic record in any event. Joint Br. at 9. Each cited instance merely reflects the lexicography’s teaching that a node “refers to a processing unit or subunit.” Nor can NVIDIA documents—extrinsic evidence that post-dates the patent’s alleged priority date by over 10 years and which describe a different computing architecture—impact ACS’ definition of “node.” ACS’ lexicography controls. *Advanced Fiber*, 674 F.3d at 1372.

3. Plaintiff's Reply Position

The term “node” appears in quotes three times. The “background” section states:

Computer clusters include a group of two or more computers, microprocessors, and/or processor cores (“**nodes**”) that intercommunicate so that the nodes can accomplish a task as though they were a single computer.

Ex. B, 1:19–23. The “detailed description of preferred embodiments” section states:

In one embodiment, a computer system comprises one or more processors such as, for example, a microprocessor that can include one or more processing cores (“**nodes**”). *The term “node”* refers to a processing unit or subunit that is capable of single-threaded execution of code.

Id., 4:43–47. NVIDIA argues that the third instance expressly defines “node.” Considering the specification as a whole reveals that this reference uses “node” narrowly in the context of a specific embodiment, in contrast to the more general usage of “node” earlier. “Node” generally refers to computers, microprocessors and/or processor cores. *Id.*, 1:19–23. Use of “refers to,” in one example, is not sufficient to define a term. *See Parkervision, Inc. v. LG Elecs., Inc.*, 2022 WL 2240465, at *6, *8 (W.D. Tex. June 21, 2022).

Neither expert expressly opined on “node” because NVIDIA identified “node” for construction eight months after expert reports were served. However, Dr. Foster’s report is consistent with ACS’s construction, and he agreed that a node “can be a single microprocessor or a computer.” Ex. E, ¶ 25.

4. Defendants’ Sur-Reply Position

The specification plainly defines “node”: “The term ‘node’ refers to a processing unit or subunit that is capable of single-threaded execution of code.” ’768 at 4:45–47. Although that statement occurs in a section titled “Detailed Description of Preferred Embodiments,” nothing in the statement itself limits the definition to only a single embodiment. Moreover, ACS fails to point to any embodiment where the specification’s definition does not apply. Finally, the earlier

statement cited by ACS—which refers to “nodes” as being able to take the form of “computers, microprocessors, and/or processor cores”—does not negate the later definition. Instead, both are consistent: the processing unit or subunit that is capable of single-threaded execution of code may take the form of a computer, microprocessor, or processor core. Foster Decl., ¶ 25.

C. “Single-Node Kernel” (claims 1, 26, 29, 35)

ACS	Plain and ordinary meaning: program code that can run on one node
NVIDIA	a kernel that is designed to communicate with and run on only a single node

1. Plaintiff’s Opening Position

NVIDIA’s proposed construction is unduly narrow and not the ordinary meaning for the term “single-node kernel.” A POSITA would have understood the ordinary meaning of “single-node kernel” as program code that can run on one node. *See* Ex. D, ¶¶ 102–04, 109. This ordinary meaning is consistent with the claims’ usage of “single-node kernel,” which is largely coextensive with their use of “kernel.” *See supra* § II.A.1; Ex. D, ¶ 102. This ordinary meaning also is consistent with the specification. Ex. B, 2:58–3:4, 3:13–26, 4:14–18, 29:66–30:1; Ex. D, ¶¶ 103–05.

As with “kernel,” NVIDIA’s construction attempts to improperly import limitations from the specification. NVIDIA’s construction is an amalgamation of two separate portions of the specification: “a kernel that is designed to communicate with a single node,” Ex. B, 1:37–38, and “a software module designed to run on a single node,” *id.*, 2:18–19. *See* Ex. D, ¶ 106. But neither of these passages support combining them as requirements as NVIDIA does, and the addition of “and run on *only* a single node” is nowhere found in the specification. *See id.*

These sections are neither definitional nor disavowals; they do not even refer to “single-node kernels.” Ex. B, 1:37–43, 2:18–19; Ex. D, ¶ 106. Their language—“[a]n example” and

“[o]ne embodiment”—further precludes their “clearly set[ting] forth a definition” or evincing a “clear disavowal” of claim scope. Ex. B, 2:18; Ex. D, ¶ 106; *Thorner*, 669 F.3d at 1365.

NVIDIA’s construction also would exclude Mathematica kernels, a disclosed single-node kernel embodiment. *See e.g.*, Ex. B, 11:40–45 (“kernel modules ... for example, Mathematica kernels”); *see also id.*, Cl. 2. Mathematica kernels do not necessarily “communicate with and run on only a single node.” Ex. M at -801 (“[I]t is not uncommon to run the front end for *Mathematica* on one computer, while running the kernel on a quite separate computer.”), -782, -799–800. NVIDIA’s attempt to import narrowing limitations inconsistent with the specification is improper.

2. Defendants’ Answering Position

Because the term “*single-node* kernel” was coined by ACS and thus has no “accepted meaning in the art,” the term should not be construed “broader than the disclosure in the specification.” *Indacon, Inc. v. Facebook, Inc.*, 824 F.3d 1352, 1357 (Fed. Cir. 2016). The ’768 patent specification explains that the problem being solved is extending cluster computing capability to software with kernels that would otherwise be limited to a single node in two ways. First, single-node kernels are “designed to communicate with a single node.” ’768 at 1:37–43 (“Some software programs include a kernel that is designed to communicate with a single node. An example of a software package that includes a kernel that is designed to communicate with a single node is Mathematica®....”); Foster Decl., ¶ 67. Second, single-node kernels are designed to “run on only a single node.” ’768 at 1:22–29, 2:18–21, 4:56–59; Foster, ¶¶ 64–66.

NVIDIA’s construction thus properly makes clear the claimed “*single-node* kernel” is designed to communicate with and run on only a single node, as would be understood by a POSITA reading the claims and written description. Foster Decl., ¶¶ 64–69.

ACS, on the other hand, proposes that a “single-node kernel” can be—but is not required to—run on a single-node. Such a construction effectively reads out the claim term “single-node” because it includes kernels that are designed to communicate with and run on *multiple* nodes—the opposite of a *single*-node kernel. Foster Decl., ¶¶ 64–69. Confirming its intent to read out the “single-node” portion of the term, ACS asserts that “the claims’ usage of ‘single-node kernel’ ... is largely coextensive with their use of ‘kernel.’” Joint Br. at 13. Contrary to ACS’ argument, the phrase “single-node” is not mere surplusage that can be ignored. ACS also argues that the Court can ignore the passages cited by NVIDIA because they “are neither definitional nor disavowals.” (Joint Br. at 13.) ACS’ argument misreads *Thorner*, especially where, as here, “single-node kernel” is a coined term without any plain and ordinary meaning. *Indacon*, 824 F.3d at 1357–58 (“[W]e need not find disclaimer where the specification does not permit a broader interpretation of these claim terms and the terms otherwise lack an ordinary meaning in the art.”).

Finally, ACS resorts to attorney argument and extrinsic evidence to incorrectly assert that NVIDIA’s construction would exclude Mathematica kernels. Joint Br. at 13–14. The specification expressly teaches that “a kernel that is designed to communicate with a single node is Mathematica” and that a Mathematica kernel is “designed to run on a single node.” ’768 at 1:37–43, 2:18–20. Yet, ACS argues that “Mathematica kernels do not necessarily ‘communicate with and run on only a single node.’” Joint Br. at 14. ACS’ only support is a statement from a Mathematica manual that ACS misconstrues: “[I]t is not uncommon to run the front end for Mathematica on one computer, while running the kernel on a quite separate computer.” *Id.* To the contrary, this statement is consistent with both the specification and NVIDIA’s construction in that it describes the kernel as running on a single computer, though it may be different from the computer on which the front end runs. Moreover, even if this extrinsic evidence genuinely differed

from the specification, ACS could not rely on it to “contradict the import of ... the specification.”

Vitronics, 90 F.3d at 1584

3. Plaintiff’s Reply Position

NVIDIA’s proposed construction is based entirely on the unsupported assumption that “single-node kernel” is a coined term. It is not, and NVIDIA cites no evidence to the contrary. NVIDIA’s IPR expert analyzed invalidity using the “plain and ordinary meaning” of the term. Ex. F, ¶ 42. NVIDIA’s claim construction expert also never suggested it was a coined term. *See* Ex. E, ¶¶ 64–69. NVIDIA does not argue that “node” or “kernel” are coined terms and does not explain how the word “single” transforms the combination of these words into a coined term. NVIDIA’s resulting argument that “single-node kernel” must not be construed “broader than the disclosure in the specification” is unsupported where, as here, it is not a coined term. *Indacon*, 824 F.3d at 1357 (term at issue had “no accepted meaning in the art.”).

NVIDIA baselessly asserts that ACS “reads out” the “single-node” portion of the term. ACS’s proposed plain and ordinary meaning reads out nothing. ACS’s articulation of the plain and ordinary meaning, “program code *that can run on one node*,” also recognizes “single-node.”

NVIDIA’s insertion of “*only*” into their construction forces an improper negative limitation. *IGT v. Bally Gaming Int’l, Inc.*, 659 F.3d 1109, 1116–17 (Fed. Cir. 2011) (declining to construe “one” as “only one”). It also improperly excludes a preferred embodiment, Mathematica. Ex. B, 2:18–20. A Mathematica kernel, which is an exemplary single-node kernel, *id.*, 2:18–20, may communicate with a node other than the node on which it runs. The specification also discusses how slave Mathematica kernels communicate with a master Mathematica kernel on a separate node when used in gridMathematica. *Id.*, 1:54–59. NVIDIA’s construction is therefore

inconsistent with the specification because a Mathematica kernel “is designed to communicate with” a different node than the node on which the kernel runs.

4. Defendants’ Sur-Reply Position

ACS first asserts “single-node kernel” is not a coined term, but fails to cite *any* use of that term *anywhere* outside the ’768 patent. Joint Br. at 16. And ACS does not dispute that, for a coined term, the correct construction cannot be broader than the disclosure in the specification. *Id.*

ACS next argues that the term “single” should not be construed as “only one,” relying on *IGT*. Joint Br. at 16 (citing *IGT v. Bally Gaming Int’l, Inc.*, 659 F.3d 1109, 1116–17 (Fed. Cir. 2011)). But *IGT* supports NVIDIA’s construction, because the Court required that a particular command “be issued to a *single* gaming device” because the word “‘one’ modifies devices that will receive a particular command.” 659 F.3d at 1116–17 (emphasis added). Similarly here, because “single-node” modifies “kernel,” such a kernel must be limited to “only one” node, not one or more nodes.

Finally, ACS’ gridMathematica argument is mistaken, because gridMathematica is not an embodiment of the claimed invention. Instead, it is distinguished by the patent as outside the claim scope. ’768 at 1:43–62. Second, gridMathematica adds its functionality to Mathematica kernels, which are single-node kernels within the scope of NVIDIA’s construction, because they are designed to communicate with and run on only a single node, as explained previously. Joint Br. at 14.

D. “Third Node” Phrase (claims 1, 26, 29)

ACS	Plain and ordinary meaning: “wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node”
-----	--

NVIDIA	This limitation requires a precise order of operations involving three specific nodes: (1) the third node receives, from the second node, a result of a calculation that was performed by the second node in a different claim limitation; (2) the third node performs a different calculation based on the received result; and (3) the third node sends the new result to the first node.
--------	---

1. Plaintiff's Opening Position

NVIDIA's proposed construction is not the plain and ordinary meaning for this phrase. A POSITA would have understood the phrase uses terms that require no further construction beyond their ordinary meaning. Ex. D, ¶¶ 111, 115. Nothing suggests the collective meaning of the "Third Node" Phrase is different from the actual claim language itself. *Id.*

Neither side genuinely questions the meaning of these claim terms. NVIDIA all but concedes the sufficiency of the "Third Node" Phrase's plain and ordinary meaning, as NVIDIA did not attempt to construe any terms within this limitation, apart from "node(s)." NVIDIA also did not propose a similar construction for the similar claim limitation in Claim 35, instead insisting that limitation would not have reasonably informed a POSITA of the scope of the claim, contrasting the "precise order of operations" purportedly required here. *See infra* § II.H.2.

NVIDIA appears to make an argument resembling disavowal, relying on statements made in IPRs of the '768 Patent. *See* Ex. N at 26–28; Ex. O at 18–21; *see also* Ex. D, ¶ 112. But, "in order for prosecution disclaimer to attach, the disclaimer must be both clear and unmistakable.... Where the alleged disavowal is ambiguous, or even amenable to multiple reasonable interpretations, [Courts] ha[ve] declined to find prosecution disclaimer." *Mass. Inst. of Tech. v. Shire Pharms., Inc.*, 839 F.3d 1111, 1119 (Fed. Cir. 2016); *see also Thorner*, 669 F.3d at 1365.

NVIDIA attempts to rely on statements that were not limiting claim scope to avoid prior art, but criticizing the evidentiary showings of NVIDIA's IPR petitions. *See Cutsforth, Inc. v. Westinghouse Air Brake Techs. Corp.*, No. 17-1025, 2018 WL 4076837, at *8–9 (W.D. Pa. Aug.

27, 2018) (“this is not a scenario in which a patent owner limited the scope of a claim during prosecution to gain patent allowance ... Plaintiff’s argument attempted to ... show that PTAB’s lack of analysis was contrary to law.”); Ex. D, ¶¶ 112–13. For example, ACS argued “the Petition refers to the prior art’s general disclosure that multiple nodes may be involved in evaluating mathematical expressions and then speculates that the prior art could work in [the claimed] manner” and “[i]t is clear on the face of Petitioner’s argument that Figure 5 does not actually disclose every detail about the claim limitation and that Petitioner is merely speculating about how Figure 5 allegedly could work.” Ex. N at 27–28; Ex. O at 19–20. The PTAB similarly rejected the *proof* supporting NVIDIA’s arguments. See Ex. P at 26 (“based on speculation of what may happen”), 28 (“Petitioner has not provided sufficient objective evidence”), 30 (same); Ex. Q at 27 (same), 29 (same). ACS’s statements were not clear disavowals of claim scope, but argument paraphrasing to highlight failures of proof in NVIDIA’s petitions. See *Cutsforth*, 2018 WL 4076837, at *8–9; Ex. D, ¶¶ 112–13. For the same reason, this is an alternate reasonable interpretation of ACS’s statements, which precludes a finding of disavowal. *Mass. Inst.*, 839 F.3d at 1119.

Further, NVIDIA’s “construction” is also improper for several reasons. First, NVIDIA removes the critical claim language that the third node is “configured to” perform the recited operations, which converts claimed capability into required operation. *INVT SPE LLC v. Int’l Trade Comm’n*, 46 F.4th 1361, 1372–74 (Fed. Cir. 2022). Second, NVIDIA unjustifiably converts this apparatus claim into a series of method steps within an apparatus claim. *Hewlett-Packard Co. v. Bausch & Lomb Inc.*, 909 F.2d 1464, 1468 (Fed. Cir. 1990) (“apparatus claims cover what a device *is*, not what a device *does*”); see also *UltimatePointer, L.L.C. v. Nintendo Co.*, 816 F.3d 816, 826 (Fed. Cir. 2016). NVIDIA has argued that such mixed claims are invalid under 35 U.S.C.

§ 112. Ex. H at 122. Third, NVIDIA’s construction adds unsupported language (*e.g.*, the third node “performs a different calculation”). Fourth, NVIDIA’s construction is syntactically incorrect because it is self-referential (*e.g.*, “this limitation”) and references other limitations rather than structure (*e.g.*, “a different claim limitation”). Ex. D, ¶ 114.

2. Defendants’ Answering Position

The terms at issue each require that the “third node is configured to” perform specific operations in a precise order. ’768 at claims 1, 26, 29. Despite this clear claim language, ACS argues that the term requires only mere “capability.” Joint Br. at 19.

The Federal Circuit distinguishes “configured to” limitations from “capable of” limitations. For example, in *Typhoon Touch*, the Federal Circuit affirmed a construction of “a memory for storing at least one data collection application configured to determine contents and formats of said inquiries” that required “a memory that *must perform* the recited function” and rejected the plaintiff’s argument that “it suffices if the memory is capable of being configured to store data collection applications, even if the memory is not so configured.” *Typhoon Touch Techs., Inc. v. Dell, Inc.*, 659 F.3d 1376, 1380–81 (Fed. Cir. 2011) (emphasis added); *Aspex Eyewear, Inc. v. Marchon Eyewear, Inc.*, 672 F.3d 1335, 1349 (Fed. Cir. 2012) (equating “adapted to” with “configured to” and finding that “adapted to” required more than mere capability).

While ACS cites the Federal Circuit’s *INVT* decision (without discussion) as purportedly supporting its position, just the opposite is true. Joint Br. at 19 (citing *INVT SPE LLC v. Int’l Trade Comm’n*, 46 F.4th 1361, 1371 (Fed. Cir. 2022)). The Court recognized the “distinction between configuration-type and capability-type claims.” *Id.* (citing *ParkerVision, Inc. v. Qualcomm Inc.*, 903 F.3d 1354, 1361 (Fed. Cir. 2018) (“our cases distinguish between claims with language that recites capability, and those that recite configuration”)). The Court also cited its

earlier decision in *Bell Aerosol*, noting that it “construed the *Bell Aerosol* claim to be a configuration-type claim, which required showing that the candle holder *was actually placed* on its cover.” *Id.* at 1372 (citing *Ball Aerosol & Specialty Container, Inc. v. Ltd. Brands, Inc.*, 555 F.3d 984, 995 (Fed. Cir. 2009)) (emphasis added). Moreover, even for capability-type claims, as in *INVT*, the Federal Circuit still requires “a device with the capability of performing the recited functions when in operation *without any modification or further programming*.” 46 F.4th at 1374 (emphasis added). This is in contrast to the merely “capable of” without “required operation” position advocated by ACS. Joint Br. at 19.

District courts likewise refuse to equate “configured to” claim limitations with mere capability. As this Court recently recognized: “In *Aspex Eyewear, Inc. v. Marchon Eyewear, Inc.*, the Federal Circuit indicated that ‘configured to’ has a narrower definition than ‘having the capacity to’ or ‘capable of.’ And district courts have relied on *Aspex Eyewear* to construe ‘configured to’ as ‘programmed to.’” *Acuity Brands Lighting, Inc. v. Ultravision Techs, LLC*, 19-cv-2207-MN, 2021 WL 3187439, at *7 (D. Del., Jul. 28, 2021) (construing “configured to” as “designed to”). In *Radware, Ltd. v. A10 Networks, Inc.*, the court construed “wherein the network controller is further configured to translate a source IP address of the server” as “programmed to [perform certain functions].” 13-cv-2024, 2014 WL 1572644, at *12–13 (N.D. Cal. Apr. 18, 2014), *aff’d* 697 Fed. App’x 700, 701 (Fed. Cir. 2017). The court further stated that “the claimed feature [must] be included in the software” and that “merely being ‘capable of’ performing a function is not enough.” *Id.* at *12.² Under this case law, the Court should refuse to construe the “third node” limitation as requiring mere capability.

² The four asserted independent claims all recite other “configured to” limitations. Other than the “third node” and the “configured to access a non-transitory computer-readable medium...”

Furthermore, ACS added the “first,” “second” and “third” node limitations during prosecution to overcome a prior art rejection. ACS argued “the cited references do not disclose the *combination of features* recited in the claims as amended.” Ex. C, at 81–82 (emphasis added). ACS therefore treated the three “node” limitations—each of which includes the “configured to” language—as a “combination of features” rather than mere capabilities. *See* Ex. C, at 100 (NOA).

Moreover, when faced with the likelihood that its ’768 patent would be invalidated in the recent IPRs, ACS disclaimed any construction of the “third node” limitation that does not require a device to actually be configured to perform the “precise order of operations” specified in that limitation to practice the claim.

In response to NVIDIA’s petitions for *Inter Partes* Review, ACS argued that the claims recite “a precise order of operations” that overcomes prior art. ACS told the PTAB this “precise order” is a “fundamental improvement” over the prior art. Ex. O, at 9, 18–19. ACS stated this “precise order” is set forth in the “third node” limitation:

This limitation recites a precise order of operations involving three specific nodes: (1) the third node receives, from the second node, a result of a calculation that was performed by the second node in a different claim limitation; (2) the third node performs a different calculation based on the received result; and (3) the third node sends the new result to the first node.”

Id. at 18–19; Ex. N, at 26–27 (same). ACS repeated this statement and relied upon it to distinguish prior art: “The Petition fails to prove its allegation that the prior art discloses this detailed limitation. To prove that allegation, the Petition would need to show where the prior art teaches the precise order of operations, involving three specific nodes, specified by the claim.” Ex. O, at 19; Ex. N, at 27 (same).

limitations that are being briefed herein, ACS did not ask the Court to construe any other “configured to” limitation as requiring mere capability. ACS has thus waived any such argument.

The PTAB adopted ACS' position regarding claim scope as the *sole* reason to deny institution: "We agree with Patent Owner that [the independent claims] require[] a precise order of operations involving the recited first, second, and third nodes." Ex. P, at 26.

NVIDIA thus proposes that the claim construction for this term include the exact language ACS argued to the PTAB and the PTAB relied on in denying NVIDIA's IPR petitions. *See Aylus Networks, Inc. v. Apple, Inc.*, 856 F.3d 1353, 1362 (Fed. Cir. 2017) (statements made in *inter partes* review can "support a finding of prosecution disclaimer").

ACS argues that its statements to the PTAB should be ignored because they were ambiguous and made only to attack NVIDIA's supposed failure of proof. Joint Br. at 18-19. But ACS does **not** cite the actual statements NVIDIA relies on above. Those statements demonstrate ACS first argued that the claim scope required a "precise order of operations" and *then* relied on that disclaimer to cast doubt on the prior art.

ACS also raises four quibbles with respect to its own language from the IPR proceedings. Joint Br. at 19–20. First, it raises the "capability" argument addressed above. *Id.* Second, ACS argues that "NVIDIA unjustifiably converts this apparatus claim into a series of method steps within an apparatus claim." *Id.* This argument fails, because the language in NVIDIA's construction faithfully tracks ACS' actual statement to the PTAB (which the PTAB adopted) and simply refers to functions the third node must be configured to perform. Third, ACS asserts "NVIDIA's construction adds unsupported language (*e.g.*, the third node 'performs a different calculation')." *Id.* at 20. If ACS would like to modify its previous statement to more precisely track the claim language by saying "performs a different mathematical expression evaluation," NVIDIA would be amenable to that change. '768 at 30:45–57. Finally, ACS argues that NVIDIA's construction is incorrect because it is self-referential and refers to the preceding claim

limitation. Joint Br. at 20. Once again, NVIDIA's construction precisely tracks ACS' actual statement to the PTAB (which the PTAB adopted) and ACS doesn't cite any authority precluding a "self-referential" portion of a claim construction.

Accordingly, NVIDIA requests that the Court provide a construction that includes the language ACS used to overcome prior art in the IPR proceedings and reject ACS' effort to rewrite the claim as requiring mere "capability."

3. Plaintiff's Reply Position

a. NVIDIA fails to show clear and unequivocal disclaimer

NVIDIA's construction is not commensurate with ACS's alleged disclaimer of "any construction of the 'third node' limitation that does not require a device to actually be configured to perform the 'precise order of operations' specified in that limitation." NVIDIA's construction does not limit ACS to a third node actually configured to perform a precise order of operations. NVIDIA instead removes "configured to" entirely, thereby improperly converting this limitation into a method step. NVIDIA is well aware that mixed apparatus-method claims are invalid, *see* Ex. H at 122, and its proposed construction would therefore improperly invalidate claim 35.

There was no *clear* disavowal. NVIDIA eliminates context relevant for ACS's statements. ACS argued that "[t]he Petition does not allege that this limitation would have been obvious; it alleges that the prior art *discloses* the limitation. ... the Petition refers to the prior art's *general disclosure* that multiple nodes *may be involved* ... and then *speculates* that the prior art could work" Ex. N at 27; Ex. O at 19. ACS was not distinguishing a "precise order" from a reference's non-precise order, *e.g.*, with preceding or intervening steps, but rather paraphrasing to highlight that NVIDIA's speculation did not meet the bar for anticipation. The PTAB acknowledged ACS's statements focused on NVIDIA's failure of proof, Ex. P at 25; Ex. Q at 25,

and then dismantled NVIDIA's arguments as speculative, Ex. P at 26, 28, 30; Ex. Q at 27, 29. NVIDIA focuses on the PTAB's alleged "adopt[ion] [of] ACS position." Ex. P at 26. However, the entire IPR context clarifies that the PTAB simply underscored NVIDIA's failure to prove the limitation was disclosed. *Id.* (quoting limitation's language). The PTAB did not make this "adopt[ion]" in the other IPR of this patent, which addressed this same limitation. *See* Ex. Q at 25–29. ACS also never claimed that this limitation was a "fundamental improvement." *See* Ex. O at 1–2.

NVIDIA's arguments based on the prosecution history are ambiguous. These statements about an unspecified "combination of features" are unclear. *See* Ex. C at 81–82. It was plainly the "combination of *all limitations*" that made the claims allowable. *Id.* at 100. NVIDIA's expert does not rebut ACS's expert. The Court should give this claim its plain and ordinary meaning.

b. NVIDIA's "configured to" arguments do not support its construction

NVIDIA's focus on "configured to" with respect to this term is puzzling given NVIDIA's position. In lieu of a construction, NVIDIA offers a comment, which allegedly "precisely tracks ACS'[s] actual statement to the PTAB." But NVIDIA's comment sheds no light on what "configured to" means. NVIDIA's extensive argument about the meaning of "configured to" does not support its proposed construction. NVIDIA's arguments are likely driven more by NVIDIA's position with respect to its belatedly identified "configured to access" term addressed below.³ *See infra* § II.I. But it is worth noting here that NVIDIA does not treat "configured to" in the "third

³ Because NVIDIA's "configured to" analysis is irrelevant here, ACS responds in the context of the "configured to access" limitation below. *See infra* § II.I.1. NVIDIA's waiver footnote turns this on its head. NVIDIA belatedly identified only one "configured to" limitation for construction. Therefore, NVIDIA waived any argument that non-identified "configured to" limitations require construction.

node” phrase the same way it treats “configured to” in the term below. To the extent that NVIDIA contends its construction construes “configured to” as “[t]his limitation requires,” that is a further inconsistency, and plainly wrong. *See infra* § II.I.1. “Configured to” should get its full plain and ordinary meaning. If asked to further articulate the meaning of “configured to” here, ACS would suggest “designed to” for the same reasons discussed below with respect to NVIDIA’s belatedly identified “configured to access” term. *See infra* § II.I.1.

4. Defendants’ Sur-Reply Position

ACS repeatedly told the PTAB the claims require a “precise order of operation.” Ex. N, at 26–28, Ex. O, at 18–20; Ex. 7, ¶¶ 66–70; Ex. 6, ¶¶ 36, 55–60. The PTAB agreed and rejected NVIDIA’s IPR petitions. Ex. P, at 26, Ex. Q, at 29–31. ACS’ reply simply fails to address the actual *disclaiming statements*. *Cupp Comut. AS v. Trend Micro Inc.*, 53 F.4th 1376, 1383 (Fed. Cir. 2022) (“The rule in *Aylus* ‘ensure[s] that claims are not argued’ by the patentee ‘one way in order to maintain [the claims’] patentability and in a different way against accused infringers.’”) (quoting *Aylus Networks, Inc. v. Apple Inc.*, 856 F.3d 1353, 1360 (Fed. Cir. 2017)).

ACS’ disclaimer underscores that the “configured to” language of this claim limitation means “programed to.” ACS attacked the IPR petition as allegedly just showing the prior art merely “could” or “may” perform the precise order of operations. Ex. N, at 26–28, Ex. O, at 18–20; *see also* Ex. 7, ¶¶ 66–70; Ex. 6, ¶¶ 36, 55–60. The PTAB agreed, holding the claims require programming with the precise order of operations, not merely the ability to be modified to perform the precise order of operations. Ex. P, at 26, Ex. Q, at 29–31.

E. “Peer-To-Peer Architecture” (claims 1, 35)

ACS	architecture in which each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node
NVIDIA	an architecture in which each node is configured to communicate with other nodes

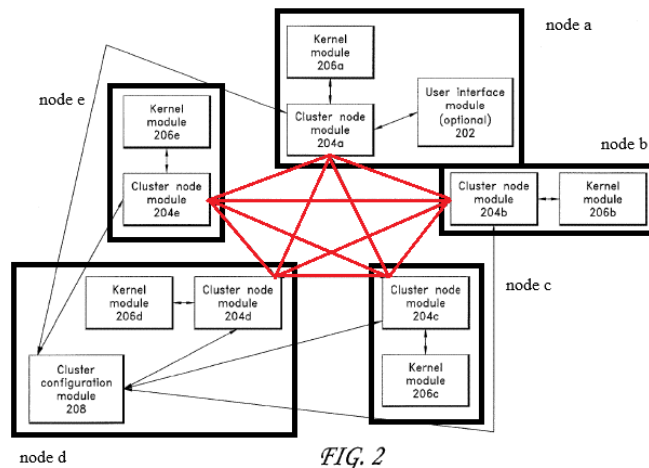
1. Plaintiff’s Opening Position

NVIDIA’s proposed construction would improperly include numerous prior art architectures expressly disclosed in the ’768 Patent. The patent describes and claims a fundamental improvement over previous efforts to allow multiple computer nodes to perform computations in parallel. Ex. D, ¶¶ 40, 50, 54–55; *cf.* Ex. B, 1:44–62, 12:30–33. It discloses an architecture to allow the nodes of a computer cluster to communicate tasks and data with one another without involving a central server or master node. Ex. B, FIG. 2, 23:13–14, 23:51–56, 24:39–40, 24:52–53, 25:1–2, 25:9–10, 25:23–24, 25:37–38; Ex. D, ¶¶ 41–42, 51–52.

The patent distinguishes “a form of grid computing known as ‘distributed computing.’” Ex. D, ¶¶ 40, 50, 54–55; *cf.* Ex. B, 1:44–62, 12:30–33. Grid computing relies on a “master node that manages a plurality of slave nodes or computational nodes.” Ex. B, 1:51–53; Ex. D, ¶ 40. The “master kernel ... handles all input, output, and scheduling of the other kernels (the computational kernels or slave kernels). Computational kernels receive commands and data only from the node running the master kernel.” Ex. B, 1:55–59; Ex. D, ¶ 40. The nodes “generally do not communicate with one another as peers.” Ex. B, 1:46–47; Ex. D, ¶ 40.

By contrast, the patent describes a “computer cluster having peer-to-peer node architecture.” Ex. B, 12:30–40; Ex. D, ¶ 41. “[E]ach cluster node module 204a-e is connected to *all* other cluster node modules” and “[t]he cluster node modules 204a-e establish communication

with one another” through “*direct connections*” between each cluster node module. Ex. B, 23:13–14, 23:51–56; Ex. D, ¶¶ 51–52. FIG. 2 shows an embodiment of the “peer-to-peer architecture.”



Ex. D, ¶ 51; Ex. B, FIG. 2 (annotated). Message passing among cluster node modules “provides the peer-to-peer behavior of the cluster node modules,” allowing them to “interact on a pair-wise or collective basis.” Ex. B, 25:22–41, 23:13–24, 24:39–44, 24:52–58, 25:1–16; Ex. D, ¶ 52. These messages can include data, *e.g.*, Ex. B, Cl. 1 (“communicate a **result** of the first mathematical expression evaluation”), as well as tasks, *e.g.*, *id.*, 6:9–25 (“MPI calls” can “parallelize program code ... and distribute **tasks** ...”). Ex. D, ¶¶ 53–55.

A POSITA would have understood that this peer-to-peer behavior is an essential feature of the “peer-to-peer architecture.” *Id.*, ¶¶ 50, 56. Unlike nodes in prior art grid computers, each node in the disclosed peer-to-peer architecture can “schedule[e]” and communicate (*e.g.*, distributing, sending, and receiving) “commands and data” with other nodes without requiring them to go through a central server or master node. *Id.*, ¶ 54; *compare* Ex. B, 1:51–59 with Ex. B, 23:13–14, 23:51–56. A POSITA would have understood that the communication of tasks and data is encompassed by “scheduling” and communicating “commands and data.” Ex. D, ¶ 54. Accordingly, a POSITA would have understood that a “peer-to-peer architecture” must

incorporate direct communication of tasks and data between nodes without requiring communications go through a central server or master node. *Id.*, ¶¶ 49–50.

NVIDIA’s construction transparently and improperly attempts to ensnare grid computing’s master-slave architecture. *Id.*, ¶ 56. By not requiring the ability to communicate both tasks and data, and allowing communications to go through a central server or master node, NVIDIA’s construction obscures the distinction between prior art grid computing and the patented cluster computing, and the fundamental feature of peer-to-peer architectures: direct communication between each node. *Id.* More generally, NVIDIA’s construction is so broad that it encompasses nearly every communication architecture, rendering this key limitation surplusage. NVIDIA’s construction does not require that each node is able to communicate with *each* other node, a defining feature of peer-to-peer architectures. *Id.* NVIDIA’s construction would encompass several topologies that NVIDIA itself distinguishes from an “all-to-all” architecture. Ex. R at 68. For example, nodes in the pictured nearest neighbor architecture communicate with the two nodes to their left and right; and nodes in the pictured tree architecture communicate with their two “branch” nodes, and their one “stem” node. *Id.* NVIDIA’s construction improperly seeks to ensnare the prior art and should be rejected.

2. Defendants’ Answering Position

“Peer-to-peer architecture”⁴ has a well-understood meaning in the art, referring to an architecture in which each node is configured to communicate with other nodes—NVIDIA’s construction. Foster Decl., ¶¶ 87–88, 90.

⁴ The term “peer-to-peer architecture” is part of the function of means-plus-function limitations in claims 1 and 35, as discussed in further detail below.

The disputes between the parties for this term turn on ACS' improper efforts to import limitations. First, the Court should reject ACS' effort to import the term "tasks and data" into its construction. Nothing in the ordinary meaning of "peer-to-peer architecture" limits communications to, or requires them to include, any particular type of content, whether "tasks," "data," or anything else. Moreover, the claims do not even use the words "tasks" or "data." Rather, claims 1 and 35 recite that mechanism using the "peer-to-peer architecture" communicates "results of mathematical expression evaluation" and "user instructions." '768 at claims 1, 35; Foster Decl., ¶ 94.

Second, the Court should reject ACS' attempt to import a negative limitation into the claim. ACS seeks to add the limitation "without the tasks and data being required to go through a central server or master node" into the term "peer-to-peer architecture." The Federal Circuit has "identified claim constructions that exclude a particular element as including a 'negative limitation' and held that such exclusions must find support either in 'the words of the claim' or through an 'express disclaimer or independent lexicography in the written description that would justify adding that negative limitation.'" *Ethicon LLC v. Intuitive Surgical, Inc.*, 847 Fed. App'x. 901, 907 (Fed. Cir. 2021); *Omega Eng'g, Inc. v. Raytek Corp.*, 334 F.3d 1314, 1322–23 (Fed. Cir. 2003). Here, ACS' argument has no anchor in the words of the claim. Indeed, the terms "tasks," "data," "central server," and "master node" are not used in the claims, and "central server" is not used anywhere in the patent.

ACS likewise fails to identify any disclaimer or lexicography that could support its construction. The specification distinguishes peer-to-peer communications from "grid computing" prior art where nodes could not intercommunicate at all. '768 at 12:26–30 ("In contrast, a parallel computing toolkit (PCT) that is provided as part of the gridMathematica software package does

not provide a means for instances of the same code running on different nodes to communicate.”); *id.*, 1:46–62 (distinguishing peer-to-peer architecture in which nodes “communicate with one another as peers” from a “master-slave” architecture in which a “master kernel” “handles *all* input, output, and scheduling of the other kernels”) (emphasis added); Foster Decl., ¶¶ 87–88. ACS argues that NVIDIA’s proposed construction would “ensnare” the prior-art “master-slave” architecture. To the contrary, nothing in the distinction raised in the specification requires that communications intended for a peer be routed in a particular way. For example, nothing prohibits ring or mesh routing in which nodes pass along messages intended for other nodes, hub-and-spoke routing in which messages are routed through a central node, or any other way of routing peer-to-peer communications.

ACS also claims that NVIDIA’s construction does not “require that each node is able to communicate with each other node.” Joint Br. at 29. ACS’ criticism applies equally to its own construction, which states like NVIDIA’s construction that each node communicates “with other nodes.” Moreover, here the claims themselves define the communication pattern between nodes. ’768 at 30:36–63.

To the contrary, ACS’ construction excludes a load-balancing preferred embodiment in which a single node—a master node called the “root processor”—exclusively assigns tasks to the other nodes to balance their workload. ’768 at 21:23–30; Foster Decl., ¶¶ 104–107. The fact that ACS’ construction would exclude this embodiment provides another reason for the Court to reject it. *On-Line Techs. Inc. v. Bodenseewerk Perkin-Elmer GMBH*, 386 F.3d 1133, 1138 (Fed. Cir. 2004) (“[A] claim interpretation that excludes a preferred embodiment from the scope of the claim is rarely, if ever, correct.”) (quotations omitted).

In addition, with respect to task routing, ACS cites only the following sentence: “MPI calls and advanced cluster commands are used to parallelize program code received from an optional user interface module 208 and distribute tasks among the kernel modules 206a-e.” ’768 at 6:15–19; Joint Br. at 28. Nothing in that sentence constitutes disclaimer or lexicography justifying ACS’ negative limitation. In fact, this sentence is contrary to ACS’ construction because the “distribut[ing] tasks among the kernel modules 206a-e” is done by the “master” node comprising cluster node module 204a and kernel module 206a. Foster Decl., ¶¶ 99–100 (explaining that node with user interface is a “master node”), ¶ 101 (discussing col. 6:15–19). Other portions of the specification likewise teach that tasks are required to go through a “master” node. *Id.*, ¶ 102 (discussing col. 6:19–21), ¶ 103 (discussing col. 24:35–25:21 (commands sent by “master” node to other nodes)).

The extrinsic evidence likewise supports NVIDIA’s construction. For example, the Dictionary of IEEE Standards Terms defines “peer-to-peer communication” as “communication between two or more processes or programs by which both computers can exchange data freely.” Foster Decl., ¶ 90 (Dictionary of IEEE Standards Terms, 804 (7th ed. 2000)). Nothing in that definition requires a specific type of communication routing that would exclude a central node. *Id.* Thus, the only support for ACS’ negative limitation is its expert’s opinions, which contradict intrinsic and extrinsic evidence. *Vitronics*, 90 F.3d at 1584 (“[E]xpert testimony... may not be used to vary or contradict the claim language” or “the import of other parts of the specification.”).

ACS’ citation to an NVIDIA invalidity claim chart does not support ACS’ position. Joint Br. at 29. ACS does not cite any evidence regarding how “all-to-all” communication (a term not used in the ’768 patent) relates to the claimed “peer-to-peer architecture,” but its attorney argument appears to equate the two. Joint Br. at 29. The portion of the claim chart cited by ACS shows a

slide from a 2004 presentation by named '768 inventor Dauger regarding his own *prior art* product. Ex. R, at 68. In 2008, Dauger included an *identical* slide in another public presentation describing ACS' SET product that allegedly practiced the '768 invention. Ex. 5, at 18. *Neither* slide has a communication pattern labeled as "peer-to-peer." However, one pattern is labeled "Master-Slave," and another "All-to-All." Ex. R, at 68; Ex. 5, at 18. Thus, the ACS product allegedly practicing the '768 patent used a communication pattern that ACS now argues should be *excluded* by the construction of "peer-to-peer architecture," while the prior art used a pattern that, according to ACS, *differentiates* the alleged invention. This shows that ACS' criticism of NVIDIA's construction is a misbegotten attempt to preserve validity, and nothing more.

Accordingly, the Court should reject ACS' attempt to import a negative limitation into this term.⁵

3. Plaintiff's Reply Position

ACS's construction is proper and reflects the meaning of "peer-to-peer architecture"⁶ as used throughout the specification. NVIDIA appears to concede the propriety of ACS's proposed construction in large part, agreeing that the claimed peer-to-peer architecture communicates "user instructions" and "results of mathematical expression evaluation." Apart from pointing out that "tasks" and "data" are not literally in the claims, NVIDIA gives no compelling reason why it is improper to simplify these concepts as tasks and data.

ACS's proposal that communication occur "without the tasks and data being required to go through a central server or master node" does not impose a negative limitation, but rather is

⁵ The PTAB also repeatedly rejected ACS' construction in recent IPRs. *See* Ex. 2, at 14–21, 48; Ex. 3, at 12–20; Ex. 4, at 14–24.

⁶ None of the PTAB's institution decisions addressed the "peer-to-peer architecture" term. NVIDIA's arguments are not relevant. These decisions, even if relevant, are not binding. *Adidas*, No. 1:14-CV-00130, Dkt. 201, 2 n.1 (Ex. S).

descriptive of the capability of the claimed architecture. Even if this were a negative limitation, courts have justified exclusions when “statements in the specification expressly list[] the disadvantages of using’ that element” or “distinguish[] among’ the element and alternatives.” *Novartis Pharms. Corp. v. Accord Healthcare, Inc.*, 38 F.4th 1013, 1016–17 (Fed. Cir. 2022). The specification explicitly distinguishes a peer-to-peer architecture from an architecture that requires communication with a master node. Ex. B, 1:47–53, 1:46–62. Disclaimer and lexicography are not applicable where, as here, ACS’s proposed construction tracks the ordinary meaning of the term in light of the specification and claims. *See, e.g., id.*, 13:15–16, 14:35–15:50, cl. 1.

NVIDIA mischaracterizes the node connected to the user interface as a “master” node. It is not a master because each node can directly intercommunicate tasks and data without involving the node connected to the user interface. *Id.*, 14:29–34, 14:48–53, 16:39–67, 17:1–19, 21:12–60. NVIDIA’s argument that ACS’s construction excludes “a preferred embodiment” fails at least because NVIDIA mischaracterizes the “root processor” as a “master node.” The “root processor” initiating the automatic load balancing is not necessarily the node connected to the user interface. It can be any node. *Id.*, 14:29–34, 14:48–53. Any of the nodes comprising an “advanced functions module 304 ... component of [a] cluster node module 204” could load balance by sending tasks to other nodes, thus emphasizing the requirement that nodes be able to communicate with all other nodes. *Id.*, 16:39–67. The specification describes many operations that may be initiated by any node, but NVIDIA frequently misconstrues these operations as only initiated by the node connected to the user interface. *Id.*, 21:12–60 (describing automatic load balancing functions within the advanced functions module on each node of some embodiments); Ex. E, ¶¶ 104–07; *see also* Ex. B, 17:1–19 (describing common divide-and-conquer parallel evaluation functions within the advanced functions module).

NVIDIA’s construction improperly relies on extrinsic evidence and expert testimony to provide the ordinary meaning of the term. The court should look first to the intrinsic evidence, which supports ACS’s proposed construction. *See, e.g.*, Ex. B, 13:15–16, 14:35–15:50, cl. 1. NVIDIA also ignores that ACS proposed substantially the same construction of peer-to-peer architecture in the related IPRs, despite being quick to argue disclaimer to support its construction of the “third node” phrase. *See supra* § II.D.2. ACS proposed substantially the same, narrow, construction for this term at the PTAB as it does here. Ex. N at 8; Ex. O at 8. NVIDIA does not distinguish ACS’s disavowal of claim scope. NVIDIA also does not address ACS’s observation that NVIDIA’s construction improperly ensnares numerous prior art architectures expressly distinguished in the specification. The “specification distinguishes peer-to-peer communications from ‘grid computing’ prior art where nodes could not intercommunicate at all.” *See* Ex. B, 12:26–30. NVIDIA’s construction that “each node is configured to communicate with other nodes” ignores that in grid computing, slave nodes communicated with, but only with, the master node. NVIDIA’s arguments regarding routing of messages are not relevant. The parties’ constructions relate to the logical connections between nodes. NVIDIA conflates logical communication between nodes with the physical routing of messages between those nodes, potentially by other components.

4. Defendants’ Sur-Reply Position

ACS’ construction improperly reads “tasks” and “data” into the claim, adds a negative limitation, and incorrectly excludes a preferred embodiment.

First, while ACS argues its “tasks and data” construction “simplifies” what is expressly claimed elsewhere in the claims (Joint Br. at 33), its construction would add convoluted requirements that are simply absent from the remainder of the claim. Nothing in the ordinary

meaning of “peer-to-peer architecture” requires communicating any particular type of content or excludes any type of peer-to-peer routing, and the claims do not use the words “tasks” or “data.”

Second, ACS’ argument that its “without . . . being required to” construction is not a negative limitation fails on its face because ACS explains that its purpose is to exclude peer-to-peer architectures that require any communications through a master node. Joint Br. at 33–34. ACS also does not identify any claim language that would justify excluding tasks or data from being required to go through a master node (Foster Decl., ¶ 97); the “grid computing” the specification distinguishes from peer-to-peer architecture did not permit nodes to intercommunicate *at all*. ’768 at 1:46–62, 12:26–30; Foster Decl., ¶¶ 87–88, 108–113. For the same reason, *i.e.*, because the patent-distinguished prior art that had no inter-communication between slave nodes, ACS’ “prior art ensnarement” argument also fails. Joint Br. at 34, 28.

Third, ACS mistakenly asserts that its construction would not exclude a preferred embodiment. Joint Br. at 33. While nodes of the preferred embodiments can “intercommunicate” *data*, nothing cited by ACS indicates that *tasks* are communicated by anything other than a master (*i.e.*, root) node. *Id.*; Foster Decl., ¶¶ 99–109. ACS is also wrong that the “root processor” “can be any node.” Joint Br. at 34. The specification passage cited by ACS describes mpiBcast, which broadcasts an expression “from the root processor to all the others.” ’768 at 14:29–53. This root processor is a master node; no other nodes communicate the expression (assuming, *arguendo*, that an expression is a task). Foster Decl., ¶ 100. Likewise, ACS’ statement that the nodes comprising advanced functions module 304 could load balance by sending tasks to other nodes (Joint Br. at 34) has no basis in the patent—the cited portion of the specification does not discuss either load balancing or tasks. ’768 at 16:39–67. ACS also mischaracterizes the load balancing embodiment excluded by its construction. Joint Br. at 34. In that embodiment, tasks are required to go through

a master node. Foster Decl., ¶¶ 104–107; ’768 at 21:12–60. This fact is uncontradicted—while ACS cites to ¶¶ 104–107 of its expert’s declaration, those paragraphs do not address either the load balancing embodiment or this claim limitation. Joint Br. at 34.

Finally, ACS asserts for the first time in its reply that its rejected IPR construction is a binding disavowal. Joint Br. at 35. ACS waived this argument. *See* LR 7.1.3(c)(2). ACS also contradicts itself by arguing that “[d]isclaimer and lexicography are not applicable” because “ACS’ proposed construction tracks the ordinary meaning.” Joint Br. at 34. Further, ACS cannot bind this Court to a proposed construction that the PTAB rightly rejected. In *Aylus* and *CUPP*, the Federal Circuit held only that a IPR disclaimer “can bind *the patentee* to a narrower claim interpretation in a subsequent proceeding.” *CUPP*, 53 F.4th at 1383 (emphasis added). Moreover, because the PTAB rejected it, ACS’ proposed construction did not become part of a bargain to maintain the patentability of the claims. *Aylus*, 856 F.3d at 1360. Accordingly, while ACS itself may be estopped from proposing a disputed construction different from one it proposed in IPR, this Court is not confined to ACS’ rejected IPR construction of “peer-to-peer architecture.”

F. “Cluster Node Module” (claims 4, 7, 8, 10, 27, 30, 31)

ACS	a module that establishes intercommunication among nodes in a computer cluster and allows exchanging messages among nodes using a peer-to-peer architecture
NVIDIA	a module running on each cluster node, configured to communicate messages with the single-node kernel on the same node, and with other cluster node modules

1. Plaintiff’s Opening Position

Unlike the previous terms, “cluster node module” is a coined phrase. Ex. D, ¶¶ 66–67. A POSITA would have looked to the intrinsic record to determine the meaning of this term. *Interval Licensing, LLC v. AOL, Inc.*, 766 F.3d 1364, 1375 (Fed. Cir. 2014) (“there is no uncertainty about the definitional relationship between the [coined term] and the written description”). The

specification discloses three essential characteristics of “cluster node modules,” which elucidate their definition. “Cluster node modules”: (1) establish intercommunication with one another; (2) communicate messages among themselves; and (3) exhibit “peer-to-peer behavior” through message-passing. Ex. B, 23:51–52, 24:39–40, 24:52–53, 25:1–2, 25:9–10, 25:23–34, 25:37–38; Ex. D, ¶ 69. The specification does not limit these characteristics to “one embodiment,” or otherwise suggest that they are optional, contrasting other clearly optional characteristics. Ex. D, ¶¶ 68–69. A POSITA would have understood these characteristics to be definitional. *Id.*; see *Intell. Ventures I LLC v. AT & T Mobility LLC*, 2015 WL 1393386, at *23 (D. Del. Mar. 24, 2015).⁷

NVIDIA’s construction fails to, at least, address the cluster node modules’ essential peer-to-peer behavior. NVIDIA’s construction again could coexist with prior art grid computers. See *supra* § II.E.1. A POSITA would have understood that the cluster node modules provide the cluster’s peer-to-peer behavior, and that NVIDIA’s construction of this coined term is incomplete. Ex. D, ¶ 72.

2. Defendants’ Answering Position

ACS concedes that “cluster node module” is a coined term, yet its construction leaves out critical features and adds requirements unsupported by the specification.

As discussed above, the specification describes single-node kernels, such as the Mathematica kernel, as designed to communicate with and run on only a single node. ’768 at 1:25–43. Because such single-node kernels cannot communicate with one another, they cannot operate in a computer cluster. *Id.* The alleged invention seeks to address this problem by

⁷ The PTAB did not adopt ACS’s construction of this term in instituting the now-terminated IPR of U.S. Patent No. 8,082,289. This is not binding. *Adidas AG v. Under Armour, Inc.*, No. 1:14-CV-00130, Dkt. 201, 2 n.1 (D. Del. Dec. 15, 2015) (Ex. S) (“institut[ing] an IPR is not the type of adjudication that leads to issue preclusion”). ACS respectfully disagrees with the PTAB.

interposing a software module called a “cluster node module” between single-node kernels to help implement the missing communication capabilities. As illustrated in Figure 2, each cluster node module 204 communicates with its associated kernel 206 and with other cluster node modules using messages of the MPI protocol. ’768 at 2:24–29, 5:33–55, 5:61–64, 6:3–20; Foster Decl., ¶¶ 140–43.

This repeated and consistent description of “cluster node module” is captured by NVIDIA’s construction: “a module running on each cluster node, configured to communicate messages with the single-node kernel on the same node, and with other cluster node modules.” *Irdeto Access, Inc. v. Echostar Satellite Corp.*, 383 F.3d 1295, 1300–03 (Fed. Cir. 2004) (rejecting patentee’s construction of coined term, limiting term to that “repeatedly, consistently, and exclusively” stated in the specification).

ACS’ proposed construction departs from the specification because it requires only that the “cluster node module” *establish* communication between nodes. In fact, as demonstrated above, this module is responsible for carrying out communication with messages, while the specification teaches that *another* module establishes communication. Foster Decl., ¶¶ 144–45. Moreover, ACS’ construction is overbroad because it omits the critical function of each cluster-node module communicating with its local single-node kernel, as required in the specification: “Each of the cluster node modules ... is in communication with respective kernel modules” and “[t]he cluster node module creates an illusion that a kernel module is communicating directly with the other kernel modules.” ’768 at 6:13–15, 24:29–31; Foster Decl., ¶¶ 144–45.

ACS’ construction also incorrectly adds the term “peer-to-peer architecture” to the construction of “cluster node module.” ACS has not shown that the specification limits “cluster node module” to any particular communication architecture. Rather, its role is to allow

communication between single-node kernels that are not otherwise able to communicate in a cluster. Foster Decl., ¶¶ 144–45. Moreover, where the patentee wanted to limit the claim scope to using a “peer-to-peer architecture,” it did so expressly in the claims. For example, the patentee included that limitation in claim 1, but left it out of claims 26 and 29. Finally, for claims with the “cluster node module” term that depend on claim 1, ACS’ proposal is redundant because claim 1 already includes a “peer-to-peer architecture” limitation.

3. Plaintiff’s Reply Position

NVIDIA’s construction goes much further than capturing the specification’s description of “cluster node module.” The specification never refers to the cluster node module as “a module *running on each cluster node*,” or states it must “communicate *messages with the single-node kernel on the same node*.” NVIDIA imports, and then further limits, language describing specific embodiments. NVIDIA presents no argument or evidence to support such a narrow construction.

NVIDIA’s proposed construction improperly overcomplicates a straightforward term. NVIDIA admits that “cluster node modules” are interposed between kernels “to help implement the missing communication capabilities” between the kernels. These communication capabilities are precisely the essential characteristics that ACS’s proposed construction captures. While, in some embodiments, each kernel communicates with a cluster node module, that functionality is described in the claims and need not be redundantly included in the construction of cluster node module. *See* Ex. B, cls. 6, 27, 28, 32.

NVIDIA disagrees with ACS’s proposal because NVIDIA incorrectly concludes that some component other than the “cluster node module” establishes communication between nodes. The patent imposes no such requirement. NVIDIA misinterprets the specification’s discussion of a “cluster configuration module” that “can establish communication with the cluster node modules”

and can “initializ[e] the cluster node modules.” *See* Ex. B, 11:32–40. This description does not negate that the “cluster node modules” “*establish communication with one another.*” *Id.*, 23:51–52, FIG. 4.

NVIDIA also incorrectly concludes that “using a peer-to-peer architecture” is not part of the “cluster node module.” The specification repeatedly and consistently confirms the peer-to-peer behavior of the cluster node modules. *Id.*, 5:42–44 (“each of the cluster node modules 204a-e is in communication with each of the other cluster node modules 204a-e”); *id.*, FIG. 2 (depicting peer-to-peer connections between cluster node modules); *see also id.*, 6:9–13, 25:37–42, 26:65–27:67 (“Peer-to-Peer MPI”), 28:1–45 (“Collective MPI”).

4. Defendants’ Sur-Reply Position

While ACS agrees “cluster node module” is a coined term, its construction departs from the specification. First, ACS’ attempt to limit “cluster node module” to a peer-to-peer architecture fails because the specification nowhere restricts these modules to any architecture. Foster Decl., ¶¶ 144–145. Second, the cluster configuration module, not the cluster node module, initializes the connection in the embodiment cited by ACS. ’768 at 11:32–40, 23:30–62; Foster Decl., ¶ 145. Third, ACS’ construction departs from the specification’s description of the cluster node module as the communication conduit between kernels because it does not require kernel communication. Foster Decl., ¶ 142. NVIDIA’s construction, by contrast, is fully supported. Joint Br. at 38.

Moreover, NVIDIA’s construction is not redundant of *dependent* claims. For example, claim 6 imposes additional requirements, such as the communication between a single-node kernel and cluster node module being caused by “instructions stored in a non-transitory computer-readable medium.” Claims 27, 28, 32 also include many additional limitations.

G. “Mechanism” Terms (claims 1, 26, 29, 35)

<p>“a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture” (Claim 1)</p> <p>“a mechanism for the nodes to communicate results of mathematical expression evaluation with each other” (Claim 29)</p> <p>“a mechanism to communicate results of evaluation with other computer cluster nodes using a peer-to peer architecture” (Claim 35)</p>	
ACS	<p>Not subject to 35 U.S.C. § 112, ¶ 6; no construction necessary, but should the Court require construction, then: hardware and/or software modules that support internode communication using a peer-to-peer architecture.</p> <p>Should the Court determine that § 112, ¶ 6 applies:</p> <p><u>Function:</u> communicate results of mathematical expression evaluation and user instructions between nodes</p> <p><u>Structure:</u> messaging modules that support communication using a peer-to-peer architecture</p>
NVIDIA	<p>Subject to § 112, ¶ 6</p> <p><u>Function:</u> “communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture” (claim 1); “communicate results of mathematical expression evaluation with each other” (claim 29); “communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture” (claim 35)</p> <p><u>Structure:</u> Message-Passing Interface (“MPI”) module 302 RMQ received message queue 306 and MRQ message receiving queue 308 configured to execute the process described at 25:29-43.</p>
<p>“a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using asynchronous calls” (Claim 26)</p>	
ACS	<p>Not subject to 35 U.S.C. § 112, ¶ 6; no construction necessary, but should the Court require construction, then: hardware and/or software modules that support asynchronous internode communication using a peer-to-peer architecture.</p> <p>Should the Court determine that § 112, ¶ 6 applies:</p> <p><u>Function:</u> communicate results of mathematical expression evaluation and user instructions between nodes</p> <p><u>Structure:</u> messaging modules that support asynchronous communication using a peer-to-peer architecture</p>
NVIDIA	Subject to § 112, ¶ 6

	<p>Function: communicate results of mathematical expression evaluation with each other using asynchronous calls</p> <p>Structure: Message-Passing Interface (“MPI”) module 302, including mpiISend, mpiIRecv, and mpiTest commands as described at 14:1-28, RMQ received message queue 306 and MRQ message receiving queue 308 configured to execute the process described at 25:29-43.</p>
--	---

1. Plaintiff’s Opening Position

The “Mechanism” terms are not means-plus-function limitations subject to § 112, ¶ 6. These terms need no construction as they would have been readily understood by a POSITA.

a. Section 112, ¶ 6 does not apply.

When a limitation does not use “means,” it is presumed to not be a means-plus-function limitation; NVIDIA must prove otherwise. *See Zeroclick, LLC v. Apple Inc.*, 891 F.3d 1003, 1007 (Fed. Cir. 2018). None of these limitations use “means,” thus § 112, ¶ 6 is presumed **not** to apply. NVIDIA cannot meet its burden to “demonstrate[] that the claim term fails to recite sufficiently definite structure or else recites function without reciting sufficient structure for performing that function.” *Id.* The terms would have been understood, in light of the intrinsic record, as connoting definite structure to a POSITA: hardware and/or software modules that support internode communication using a peer-to-peer architecture (Claims 1, 29, and 35). Ex. D, ¶¶ 118, 132, 154. For Claim 26, the internode communication is asynchronous. *Id.*, ¶¶ 161, 169.

“Mechanism” is used several times in the independent claims. Claims 1 and 26 recite “a **mechanism** for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture,” “communicate at least some of the user instructions using the **mechanism** ...,” and “after communicating at least some of the user instructions using the **mechanism**.” Ex. D, ¶ 118. Claims 29 and 35 are similar. *Id.*, ¶¶ 153, 161. A POSITA would have understood that the mechanism could communicate both results and user instructions. *Id.*, ¶¶ 119, 151, 161. The peer-to-peer architecture recited in Claims 1 and 35 is yet more structure

for the internode communication.⁸ *Id.*, ¶¶ 119, 151; *see also supra* § II.E. Claim 26 recites more structure by requiring communication through asynchronous calls, describing the types of calls used to communicatively connect the nodes. Ex. D, ¶ 161. Thus, the claims describe a communication architecture, which specifies both how the mechanism communicates and what the nodes communicate. In view of this, a POSITA would have understood the limitations to invoke hardware and/or software modules that support internode communication using a peer-to-peer architecture, and that for Claim 26 internode communication is asynchronous, and that these communications include evaluation results and user instructions. *See* Ex. D, ¶¶ 119, 152, 161. Dependent claims provide more structural support for the “Mechanism” terms used in the independent claims. *Id.*, ¶ 120. Claim 3 recites that “the mechanism comprises a message passing interface,” which is specific structure consistent with a POSITA’s understanding. *Id.* Claims 27 and 28 detail the contents of the asynchronous calls. *Id.*, ¶¶ 162–63.

In *Zeroclick*, the Federal Circuit held that it was error to treat “program” and “user interface code” as nonce words subject to § 112, ¶ 6 because the district court “removed the terms from their context, which otherwise strongly suggest[ed] the plain and ordinary meaning of the terms.” 891 F.3d at 1008. Here, a POSITA would have known that the term “mechanism” is not used alone and generically, but rather, in context, as a specific reference to conventional internode messaging hardware and software existing in the art at the time of the inventions. Ex. D, ¶¶ 119, 152, 161. Likewise, in *Personalized Media*, the Federal Circuit held it was error to treat “digital detector” as a means-plus-function limitation. *See Personalized Media Commc’ns, LLC v. Int’l Trade Comm’n*, 161 F.3d 696, 704 (Fed. Cir. 1998). Even though the term did not “specifically evoke a

⁸ NVIDIA concedes the “Mechanisms” terms share structure and that Claim 26 only *adds* structure. Ex. A at 13–16. Thus, ACS treats both “Mechanism” terms as using the peer-to-peer architecture.

particular structure, it d[id] convey to [a POSITA] a variety of structures known as ‘detectors.’” *Id.* at 705. Here, the mechanism limitations would have conveyed to a POSITA a variety of hardware and/or software modules that support internode communication using a peer-to-peer architecture, and that for Claim 26 the internode communication is asynchronous, and that these communications include evaluation results and user instructions. Ex. D, ¶¶ 119, 152, 161.

Reading the claims in light of the specification further reinforces this understanding. *Id.*, ¶¶ 121–25, 151, 164–67. The specification depicts an embodiment of the claimed peer-to-peer architecture. Ex. B, FIG. 2; Ex. D, ¶ 121. The specification describes how “each of the cluster node modules ... communicat[es] with each of the other cluster node modules” and how “[c]ommunications can occur between any two or more cluster node modules [] and not just between ‘adjacent’ kernels.” Ex. B, 5:56–6:13, 5:35–44; Ex. D, ¶ 122. The specification thus provides an example of the peer-to-peer architecture over which internode communication is supported. Ex. D, ¶¶ 121–22.

The specification describes the “Cluster Node Module Operation” as using the mechanism for node-to-node and collective communication: “[t]his process provides the peer-to-peer behavior of the cluster node modules 204a-e. *Via this mechanism*, code running within multiple, simultaneously running kernel modules ... can interact on a pair-wise or collective basis” Ex. B, 24:35–25:43; Ex. D, ¶ 125. The communication modules use the peer-to-peer architecture.

The specification goes on to describe an embodiment of the cluster node module that includes an “MPI module.” The MPI module can implement basic MPI calls for sending data, equations, formulas, and/or other expressions from one node to another, and collective MPI calls to provide multi-node data movement. Ex. B, 12:41–46; Ex. D, ¶ 123. An MPI module is an example of a hardware and/or software module that supports internode communication, for Claim

26 asynchronous communication, through messaging, *e.g.*, basic and collective messaging. Ex. D, ¶¶ 123–25, 165–166. The specification explicitly links the “mechanism” to the MPI module messaging by stating that “[c]ollective calls can also provide commonly used *mechanisms* to send expressions between groups of nodes.” Ex. B, 14:35–43 (emphasis added); Ex. D, ¶ 124. These disclosures would have further indicated to a POSITA that the “Mechanism” terms are hardware and/or software modules that support internode communication using a peer-to-peer architecture, and that for Claim 26 the internode communication is asynchronous, and that these communications include evaluation results and user instructions. Ex. D, ¶¶ 124–25, 154, 164.

The prosecution history confirms that § 112, ¶ 6 does not apply. During prosecution, the Examiner issued a § 112, ¶ 6 rejection but not due to the “Mechanism” terms. Ex. C at 2 (Preliminary Amendment at 2), 8–11 (Non-Final Rejection at 4–7); Ex. D, ¶¶ 125–27. The Examiner did not apply § 112, ¶ 6 to the “Mechanism” terms as he understood the structure they invoked. Ex. C at 8–11 (Non-Final Rejection at 4–7); Ex. D, ¶ 127. NVIDIA’s attempted IPRs of the ’768 Patent further confirm this. Neither party proposed a construction for the “Mechanism” terms or asserted that § 112, ¶ 6 applied. *See* Ex. T at 14; Ex. U at 16–17; Ex. N at 6–16; Ex. O at 6–15; Ex. D, ¶¶ 131, 154, 168. Indeed, NVIDIA’s IPR expert did not assert § 112, ¶ 6 applied to the “Mechanism” terms and he understood them to have their plain and ordinary meaning when opining on invalidity. Ex. F at 24, 40; Ex. G at 24–25, 77; Ex. D, ¶¶ 131, 154, 168.

Numerous cases have applied the presumption against means-plus-function treatment when analyzing “mechanism” language and subsequently found the presumption not rebutted. *See, e.g., Midwest Athletics and Sports Alliance LLC v. Xerox Corp.*, 2020 WL 7692767, at *14–15 (W.D.N.Y. Dec. 28, 2020) (marking mechanism); *Extang Corp. Undercover, Inc. v. Truck Accessories Grp., LLC*, 2020 WL 6888277, at *6–7 (D. Del. Nov. 24, 2020) (release mechanism);

Unicorn Glob. Inc. v. Golabs, Inc., 2020 WL 2745692, at *4–6 (N.D. Tex. May 26, 2020) (rotating mechanism). NVIDIA cannot rebut the presumption that § 112, ¶ 6 does not apply.

b. If § 112, ¶ 6 applies, the Court should adopt ACS’s proposed functions.

ACS’s proposed functions are plainly recited in the claims. Ex. B, Cls. 1 (“a mechanism for the nodes to *communicate results of mathematical expression evaluation* with each other using a peer-to-peer architecture” and “*communicate at least some of the user instructions* using the mechanism”), 29 (substantially similar), 35 (“substantially similar”), 26 (“a mechanism for the nodes to *communicate results of mathematical expression evaluation* with each other using asynchronous calls” and “*communicate at least some of the user instructions* using the mechanism”). Each claim recites two functions of the mechanism: (1) communicating results of [mathematical expression] evaluations; and (2) communicating user instructions, between nodes. Ex. D, ¶¶ 134–135, 155, 170.

NVIDIA’s proposed functions are inconsistent with the claims. First, they are missing the “communicating user instructions” function entirely. *Id.*, ¶¶ 135–37, 155, 170. *See Generation II Orthotics, Inc. v. Med Tech., Inc.*, 263 F.3d 1356, 1364 (Fed. Cir. 2001) (“we decline to ... impermissibly limit the claim by adopting a function different from that explicitly recited in the claim, and such an error ... could improperly alter the identification of the structure”). Second, they misidentify structure as function. Ex. D, ¶¶ 136–137, 170. A POSITA would have understood that “a peer-to-peer architecture” and “asynchronous calls” are structure. *Id.* “Peer-to-peer architecture” and “asynchronous calls” describe structurally *how* the functions are carried out, rather than *what* function is claimed. *See BBA Nonwovens Simpsonville, Inc. v. Superior Nonwovens, LLC*, 303 F.3d 1332, 1344 (Fed. Cir. 2002). Here, “a peer-to-peer architecture” should not be included in the function for the first group of mechanism terms, and “asynchronous

calls” should not be included in the function for Claim 26’s mechanism term. *See JWW Enters., Inc. v. Interact Accessories, Inc.*, 424 F.3d 1324, 1330 (Fed. Cir. 2005) (“[t]he district court’s ... construction confuse[d] function with structure ... [r]esulting in the inappropriate inclusion of structure ... in the construction of the claimed function.”).

c. If § 112, ¶ 6 applies, the Court should adopt ACS’s proposed structures.

The specification clearly links messaging modules that support communication using a peer-to-peer architecture with the recited functions. Ex. B, 3:46–49; Ex. D, ¶¶ 139, 157, 171. These messaging modules communicate results of mathematical expression evaluation and user instructions with other messaging modules, and for Claim 26 do so asynchronously. Ex. B, 6:22–25, 16:1–21:10; 26:9–27, 12:48–51, 13:10–16, 14:26–39, 15:52–55, 16:41–42; Ex. D, ¶¶ 140, 157, 171–73. In a preferred embodiment, these messaging modules are the cluster node modules used to perform communications using a peer-to-peer architecture. Ex. B, 25:22–38, 2:30–56, 3:17–24, 4:61–62, 6:10–13; Ex. D, ¶¶ 141, 156–57, 172. In a preferred embodiment, the cluster node modules may include an MPI module that performs the communications using a peer-to-peer architecture. Ex. B, 12:41–16:39, 11:42–45; Ex. D, ¶¶ 142, 156–57, 172. The cluster node modules may communicate both tasks (*e.g.*, user instructions) and data (*e.g.*, results of expression evaluation) using a peer-to-peer architecture. Ex. B, 13:10–16, 14:36–39, 16:40–21:10; Ex. D, ¶¶ 143–44, 156–57, 172; *supra* §§ II.E–F. A POSITA would have understood “[m]essaging modules that support communication using a peer-to-peer architecture” as the disclosed structure for providing internode communication, Ex. D, ¶¶ 138–45, 156, and that communication is asynchronous in Claim 26, Ex. B, 13:41–14:34; Ex. D, ¶¶ 171–73.

NVIDIA’s proposed structures are non-starters because they proceed from NVIDIA’s incorrect function. They are also incorrect, at least, because they require parts of a preferred

embodiment, which are not clearly linked to the recited function. Ex. D, ¶¶ 145–46. The message receiving queue and received message queue are described in connection with asynchronous calls, but not peer-to-peer communications generally. Ex. B, 13:41–14:28, 25:31–37; Ex. D, ¶¶ 146–47. Thus, for Claims 1, 29, and 35’s “Mechanism” terms, this is not corresponding structure and is improperly limiting. *See Northrop Grumman Corp. v. Intel Corp.*, 325 F.3d 1346, 1354 (Fed. Cir. 2003) (holding identification of structure was erroneous where identified structure was unnecessary to perform the claimed function).

2. Defendants’ Answering Position

a. The mechanism claim terms are subject to § 112, ¶ 6

These claim terms are governed by § 112, ¶ 6, because they recite a nonce “mechanism” for performing a claimed function without the claims identifying “sufficiently definite structure” or “sufficient structure for performing that function.” ’768 at claims 1, 26, 29, 35; *Williamson v. Citrix Online, LLC*, 792 F.3d 1339, 1348–49 (Fed. Cir. 2015) (quotation omitted).

The Federal Circuit recognizes that “[g]eneric terms like ... ‘mechanism,’ are commonly used as verbal constructs that operate, like ‘means,’ to claim a particular function rather than describe a ‘sufficiently definite structure.’” *MTD Prods. Inc. v. Iancu*, 933 F.3d 1336, 1341 (Fed. Cir. 2019); *Mass. Inst. Of Tech. v. Abacus Software*, 462 F.3d 1344, 1354 (Fed. Cir. 2006) (“The term ‘mechanism’ standing alone connotes no more structure than the term ‘means.’”). Accordingly, unless an “adjective endows the claimed ‘mechanism’ with a physical or structural component,” the claimed “mechanism” is subject to § 112, ¶ 6. *Welker Bearing Co. v. PHD, Inc.*, 550 F.3d 1090, 1096 (Fed. Cir. 2008) (“Thus, the unadorned term ‘mechanism’ is simply a nonce word or a verbal construct that is not recognized as the name of structure and is simply a substitute for the term ‘means for.’”) (quotation omitted). As the Federal Circuit stated, “[w]e have never

found that the term ‘mechanism’—without more—connotes an identifiable structure.” *Media Rights Tech., Inc. v. Cap. One Fin. Corp.*, 800 F.3d 1366, 1373 (Fed. Cir. 2015).

Here, the claimed “mechanism” is not paired with *any* qualifying adjective, much less one endowing it with structure. Foster Decl., ¶¶ 122–125. This distinguishes the present case from those cited by ACS. *Midwest Athletics & Sports Alliance LLC v. Xerox Corp.*, 19-cv-6036, 2020 WL 7692767, at *14–15 (W.D.N.Y. Dec. 28, 2020) (“The addition of ‘marking’ to the otherwise generic term ‘mechanism’ adds sufficient structure to bring this term outside § 112, ¶ 6.”); *Extang Corp. Undercover, Inc. v. Truck Accessories Grp., LLC*, 19-cv-923-MN, 2020 WL 6888277, at *6–7 (D. Del. Nov. 24, 2020) (“release mechanism”); *Unicorn Glob. Inc. v. Golabs, Inc.*, 19-cv-754, 2020 WL 2745692, at *4–6 (N.D. Tex. May 26, 2020) (“rotating mechanism”).

ACS’ reliance on *Zeroclick* is inapt because it did not address the term “mechanism” and the specification made clear the claim terms at issue were “specific references to conventional graphical user interface programs or code, existing in prior art at the time of the inventions.” *Zeroclick, LLC v. Apple Inc.*, 891 F.3d 1003, 1008–09 (Fed. Cir. 2018). There is no such teaching here. ACS’ reliance on *Personalized Media* is similarly misplaced because the “digital detector” claim term “had a well-known meaning to those of skill in the electrical arts connotative of structure, including a rectifier or demodulator.” *Personalized Media v. Int’l Trade Comm’n*, 161 F.3d 696, 704–05 (Fed. Cir. 1998). “Mechanism” has no such meaning. Foster Decl., ¶ 122.

ACS argues “mechanism” connotes structure to its expert, Dr. Singh. Joint Br. at 43; Ex. D, ¶¶ 118–120. However, while Dr. Singh cites *functions* performed by the “mechanisms,” Dr. Singh’s declaration never identifies *any* actual structure (much less “sufficiently definite structure”), but instead generically refers to unspecified “hardware and/or software modules.” *Id.* This falls short as a matter of law. *Grecia v. Samsung Elects. Am., Inc.*, 780 Fed. App’x. 912,

914–15 (Fed. Cir. 2019) (“The word ‘module’ does not provide any indication of structure because it sets forth the same black box recitation of structure ... as if the term ‘means’ had been used.”).

ACS points to the phrase “peer-to-peer architecture” recited in the function of claims 1 and 35 and “asynchronous calls” recited in the function of claim 26 as alleged structure. Joint Br. at 43–44. However, the claims do *not* recite that the “peer-to-peer architecture” or “asynchronous calls” are *part of* the claimed “mechanism.” Instead, those phrases are part of the claimed *functions*, *e.g.*, in claim 1: “to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture.” Nowhere do these three claims say or even suggest that the peer-to-peer architecture or asynchronous calls are part of the mechanism—and claim 29 does not recite these terms at all—and nowhere do the claims provide *any* structure for the “mechanism” itself, much less identify “sufficient structure” in the mechanism “for performing th[e] function” of communicating mathematical expression evaluation results using a peer-to-peer architecture or asynchronous calls. *Williamson*, 792 F.3d at 1348. Moreover, even if “peer-to-peer architecture” was somehow relevant to the analysis, it does not connote sufficient structure under either party’s construction of this term. It instead refers to the generalized function of each node being configured to communicate with other nodes, as described above, and *not* to actual structures sufficient to perform the recited function. Foster Decl., ¶¶ 123–124; *Kyocera Senco Indus. Tools*, 22 F.4th at 1380 (holding that “lifter” “does not connote structure” but is instead merely “functional language”). Likewise, “using asynchronous calls” in claim 26 is purely functional language. *Id.*, ¶ 125.

ACS also cites dependent claim 3, which recites: “The computer cluster of claim 1, wherein the mechanism comprises a message passing interface.” ’768 at 31:7–8. The phrase “message passing interface” is not recited in the independent claims so it cannot provide structure for them.

Foster Decl., ¶ 22. *Grecia*, 780 Fed. App'x at 915 (applying § 112, ¶ 6 where limitations in dependent claims were “not recited in” independent claim and did not “denote a specific algorithm or other structure”). Moreover, neither ACS nor Dr. Singh explains what structure this connotes, much less whether it is sufficient to perform the function. Joint Br. at 44; Ex. D, ¶ 120. And, tellingly, ACS does not identify “a message passing interface” as structure.

In arguing that the *claim language* itself recites sufficient structure, ACS also points to portions of the *specification* describing corresponding structures. Joint Br. at 44. ACS cites portions of the specification that discuss the MPI, RMQ and MRQ modules that NVIDIA contends comprise the “corresponding structure” for the claimed means-plus-function limitation. ’768 at 12:41–53, 25:29–43. But neither ACS nor its expert contends a POSITA understands the term “mechanism” to *mean* MPI, RMQ and MRQ modules. Indeed, ACS does not limit the claim scope to those specific structures or even identify these modules as “corresponding structure” under § 112, ¶ 6. ACS’ citations thus fall short because “the specification disclos[ing] a structure corresponding to an asserted means-plus-function term does not necessarily mean that the claim term is understood by persons of ordinary skill in the art to connote a specific structure.” *MTD Prods.*, 933 F.3d at 1344. Nor does ACS contend the specification provides lexicography defining “mechanism” to mean these modules. Accordingly, “the specification does not demonstrate that the patentee intended to act as its own lexicographer and define the nonce term [mechanism] as the [MPI, RMQ and MRQ modules] of the preferred embodiment” and thus § 112, ¶ 6 applies. *MTD Products*, 933 F.3d at 1344.

ACS fails to identify anything else in the specification passages it cites (12:35–46, 24:35–25:43, 5:35–44, 5:56–6:13, Fig. 2) as constituting “sufficiently definite structure” of a “mechanism,” much less lexicography. Joint Br. at 45; Ex. D, ¶¶ 121–125. This is not surprising

as these disclosures describe how cluster node modules *function*, rather than specific structures for the “mechanism.” The absence of any such definite structure is further reflected by ACS’ recitation of legally insufficient “hardware and/or software modules” in its construction. Joint Br. at 42; *see Grecia*, 780 Fed. App’x at 914–15 (“The word ‘module’ does not provide any indication of structure.”).

ACS notes the PTO issued a § 112, ¶ 6 rejection, “but not due to the ‘Mechanism’ terms.” Joint Br. at 46. The PTO made the cited rejection for the “module” term because it found that “Applicant’s specification is devoid of sufficient disclosure of structure for these terms.” Ex. C, at 9. But the dispute here is not whether the specification discloses sufficient structure for the term-at-issue under § 112, ¶ 6 (it does, as NVIDIA describes below), but whether the use in a claim of a nonce term like “mechanism” recites sufficient structure (it does not). In fact, the examiner recognized that “‘mechanism for’” is one term in “a list of non-structure terms that may invoke § 112, ¶ 6,” (Ex. C, at 10), consistent with NVIDIA’s construction.

ACS lastly notes the parties did not construe “mechanism” in the IPRs. Joint Br. at 46. However, ACS fails to note that the PTAB expressly questioned the lack of a construction of “mechanism” under § 112, ¶ 6, which the PTAB called “a well-known nonce word,” though it did not ultimately reach that issue. Ex. P, at 21 n.20; Ex. Q, at 21 n.16. Accordingly, these terms should be construed as means-plus-function limitations.

b. The Court should adopt NVIDIA’s proposed functions.

The parties have two disputes regarding the claimed functions. First, ACS’ constructions delete words from the claimed functions. The expressly recited function of claims 1 and 35 includes “using a peer-to-peer architecture”—NVIDIA’s functions include this, while ACS’ do not. The expressly recited function of claim 26 includes “using asynchronous calls”—NVIDIA’s

function includes this, while ACS’ does not. ACS states these terms refer to structure of the “mechanism” rather than function, but that assertion fails for the reasons discussed above. In addition, ACS’ brief incorrectly describes NVIDIA’s proposed function for claim 29. That claim does not include the phrase “using a peer-to-peer architecture” and NVIDIA did not include it its proposed function.

Second, ACS’ functions include communicating “user instructions.” While this function is recited later in the claims, the claim limitations being construed here do not recite that function. However, should the Court decide to include this additional function in its constructions, it will not impact the corresponding structure because the structures that perform this additional function are the same structures that perform the remaining claimed functions, as shown below.

c. The Court should adopt NVIDIA’s proposed structures

The Court should adopt NVIDIA’s proposed structures: MPI (“Message Passing Interface”) module 302, received message queue (“RMQ”) 306, and message receiving queue (“MRQ”) 308. ’768 at 12:41–53, Fig. 3; Foster Decl., ¶¶ 128–129.

With respect to the MPI module, ACS admits “[t]he specification explicitly links the ‘mechanism’ to the MPI module messaging by stating that ‘[c]ollective calls can also provide commonly used mechanisms to send expressions between groups of nodes.’” Joint Br. at 46 (quoting 14:35–43); Ex. D, ¶ 146 (MPI module is linked to the claimed functions). There is thus no reasonable argument the MPI module is not corresponding structure. Regardless, the only structure for performing the node-to-node communication, including communicating mathematical evaluation results, disclosed in the specification includes the MPI module. ’768 at 12:41–53, 13:15–20, 14:35–15:9, 24:58–67, 6:3–8; Foster Decl., ¶¶ 129–130, 133–135.

The specification also discloses the MPI module works with the RMQ and MRQ modules to implement MPI communications. '768 at 25:29–36; Foster Decl., ¶ 131. The next two sentences that follow the discussion of using the MRQ and RMQ modules clearly links those modules to the “mechanism’s” communication function: “*This process* provides the peer-to-peer behavior of the cluster node modules 204 a-e. *Via this mechanism*, code running within multiple, simultaneously running kernel modules (for example, Mathematica kernels) can interact on a pair-wise or collective basis, performing calculations, processing, or other work on a scale larger and/or faster than one kernel could have done alone.” '768 at 25:36–43 (emphasis added); Foster Decl., ¶¶ 131–133. The combination of MPI, RMQ and MRQ is the *only* mechanism disclosed to communicate the claimed results, underscoring this structure is necessary to perform the claimed function. *Id.*

Claim 26 includes the function of “using asynchronous calls.” '768 at 32:38–40. The specification discloses that the structure corresponding to this function is the mpiISend, mpiIRecv, and mpiTest commands. *Id.* at 13:41–14:28; Foster Decl., ¶ 134.

ACS does not appear to dispute the specification links RMQ 306, MRQ 308, and the mpiISend, mpiIRecv, and mpiTest commands to the mechanism’s use of asynchronous calls in claim 26. Joint Br. at 49; Ex. D, ¶¶ 146–147. ACS does dispute that RMQ 306 and MRQ 308 are corresponding structure for claims 1, 29 and 35 on the ground that those claims do not recite asynchronous calls. Joint Br. at 49. However, the specification’s discussion of these two modules is not limited to asynchronous calls and instead extends to the “mechanism’s” communication function more broadly. '768 at 25:29–46; Foster Decl., ¶¶ 131–133, 135. For example, Fig. 3 shows both RMQ 306 and MRQ 308 as part of a “cluster node module 204 implementing MPI calls,” underscoring these queues are used for MPI communications generally, and not just

asynchronous calls. ’768 at 12:41–46. And when discussing RMQ, the specification speaks of messages generally, without limiting them to asynchronous messages: “The received message queue includes a data structure for storing messages received from other cluster node modules.” *Id.* at 21:62–66, 25:29–31. The same is true of the MRQ. *Id.* at 22:16–24.

The Court should reject ACS’ proposed corresponding structure—“messaging modules” that perform the claimed functions—because “messaging modules” is never used in the specification, has no accepted meaning in the art, and would be too generic to constitute corresponding structure even if it had been disclosed. Foster Decl., ¶ 137. *Med. Instrumentation & Diagnostics Corp. v. Elekta AB*, 344 F.3d 1205, 1210 (Fed. Cir. 2003) (structure must not only be disclosed in specification, but also “clearly link[ed]” to the recited function); *Synchronous Techs., Inc. v. Dropbox, Inc.*, 987 F.3d 1358, 1367–68 (Fed. Cir. 2021) (specification must disclose “‘adequate’ structure ... that a [POSITA] would be able to recognize,” not merely a catch-all “correspond[ing] to every known way of achieving the claimed function”). ACS’ construction rests on its expert’s misguided opinions rather than the specification. He opined that a POSITA “would not have understood the structure to be limited to any of the exemplary preferred embodiments,” but would have “understood the general references to refer to the *broader class* of messaging modules.” Ex. D, ¶ 145 (emphasis added). However, if this was in fact a POSITA’s understanding (it is not), the claims would be indefinite because disclosure of a “broad class” in lieu of specific structures does not satisfy § 112 ¶ 6. *Triton Tech of Texas, LLC v. Nintendo of Am., Inc.*, 753 F.3d 1375, 1379 (Fed. Cir. 2014) (“Disclosing the broad class of ‘numerical integration’ does not limit the scope of the claim to the ‘corresponding structure, material, or acts’ that perform the function, as required by section 112.”).

Finally, if the Court decides to include the additional function of communicating user instructions in its claim construction, the same corresponding structure identified by NVIDIA (MPI, RMQ and MRQ) also performs that function. ’768 at 25:29–46 (after discussing the use of RMQ 306 and MRQ 308, stating “[i]n this manner, user entered instructions and data specifying what work will be done via user commands can be executed more quickly and/or reliably.”)

3. Plaintiff’s Reply Position

The parties understood the plain and ordinary meanings of these terms until the PTAB speculated about their construction. Ex. T at 14; Ex. U at 16–17; Ex. N at 6–16; Ex. O at 6–15. NVIDIA’s arguments are disingenuous. The PTAB’s dicta is not controlling.

a. NVIDIA fails to rebut the presumption that § 112, ¶ 6 does not apply

Undeniably, § 112, ¶ 6 is presumed not to apply and NVIDIA must prove “by a preponderance of the evidence that [a POSITA] believes the term does not recite sufficiently definite structure.” *Dyfan, LLC v. Target Corp.*, 28 F.4th 1360, 1365–66 (Fed. Cir. 2022).

NVIDIA improperly focuses on “mechanism” and ignores other claim language. NVIDIA itself concedes that “peer-to-peer architecture” is a sufficiently definite structure that can be understood by a POSITA by proposing that it should be construed as “an architecture in which each node is configured to communicate with other nodes.” NVIDIA’s assertion that this is merely functional for purposes of its means-plus-function analysis is disingenuous. Then NVIDIA argues that “mechanism” is a nonce word and is subject § 112, ¶ 6. But NVIDIA’s own cases look at the claim as a whole for structure. *MTD Prods*, 933 F.3d at 1342 (“we do not merely consider the ... phrase [] in isolation, but look to the entire passage including functions”); *Welker Bearing*, 550 F.3d at 1096 (“[c]laim language that ... defines a ... term like ‘mechanism’ can sometimes add sufficient structure” (quoting *Abacus Software*, 462 F.3d at 1354)); *Williamson*, 792 F.3d at 1351

(analyzing function, inputs, and outputs). *Media Rights* is inapposite. ACS does not assert that “mechanism” connotes sufficiently definite structure “without more”; these terms do so when the claims are read as a whole in light of the specification. *See* 800 F.3d at 1373. NVIDIA’s improper test based on “qualifying adjectives” fails because they are not the only way to add structure. *See Dyfan*, 28 F.4th at 1366; *MTD Prods*, 933 F.3d at 1342 (analyzing functions); *Williamson*, 792 F.3d at 1351 (same). Accordingly, NVIDIA does not effectively distinguish ACS’s cases.

The terms include: a peer-to-peer architecture or asynchronous calls, connections between each node, and support for communicating expression evaluations and user instructions. Ex. B, cls. 1, 26, 29, 35. They describe inputs, outputs, and how outputs are sent. *Triplay, Inc. v. Whatsapp, Inc.*, 2016 WL 3574012, at *9 (D. Del. June 30, 2016) (holding structure including “inputs, outputs, and how [] outputs are achieved” sufficiently definite), *adopted by* 2016 WL 6778215, at *4 (Nov. 15, 2016). In context, a POSITA would have known the terms are not black boxes. Ex. B, 14:35–43 (describing “collective calls” as “**commonly used mechanisms** to send expressions”); *Zeroclick*, 891 F.3d at 1008. That *Zeroclick*, did not address “mechanism” is not compelling. *See Mad Dogg Athletics, Inc. v. Peloton Interactive, Inc.*, 2021 WL 3200994, at *13–14 (E.D. Tex. July 28, 2021) (finding a term including “mechanism” referred to a “class of known structures”). NVIDIA’s demand for more detail is not justified. “Terms defined in terms of [their] function ... not connot[ing] a precise physical structure,” may be sufficiently definite. *Personalized Media*, 161 F.3d at 705. A POSITA’s understanding of the connoted class of structures can come from the claims and the specification. *See Dyfan*, 28 F.4th at 1366.

NVIDIA baselessly asserts that ACS’s structure “falls short as a matter of law.” *Grecia* analyzed “module’s” identification of structure, not if it could be **part** of the identified structure. 780 F. App’x at 915. “Module” can be part of sufficiently definite structure. *See, e.g.,*

Parkervision, Inc. v. Hisense Co., No. 6-20-cv-00870-ADA, Dkt. 51, 66 (W.D. Tex. Aug. 29, 2022) (Ex. V) (finding § 112, ¶ 6 did not apply where “means” was not used and the sufficiently definite structure included modules). Further, even if peer-to-peer architecture and asynchronous calls are functional (they are not), they would still connote structure. *Dyfan*, 28 F.4th at 1366. The parties discussed the structure of the peer-to-peer-architecture and cluster node modules at length. *See supra* §§ II.E–F. NVIDIA cannot deny that they are sufficiently definite. Mere reference to the cluster node module’s function does not nullify its structural connotations.

NVIDIA’s cases are not persuasive. In *Kyocera*, no party claimed the term had a plain and ordinary meaning and “the surrounding claim language d[id] not describe any structur[e],” 22 F.4th at 1380. Here, peer-to-peer architecture and asynchronous calls would have had ordinary meanings that connoted structure. *Grecia* did not hold that dependent claims cannot inform the structure of independent claims. *Grecia* analyzed the dependent claims and recognized that they could. 780 F. App’x at 915. The independent claims connote structure; claim 3 confirms that structure. NVIDIA’s expert never opines on claim 3’s effect on the structure of the independent claims. *See* Ex. E, ¶¶ 119–37. NVIDIA’s erroneous citation to Ex. E, ¶ 22 is unpersuasive.

NVIDIA faults ACS for “not limiting the claim scope to those [] structures or even identify these modules as ‘corresponding structure.’” NVIDIA misunderstands § 112, ¶ 6. Corresponding structure is analyzed only if § 112, ¶ 6 applies. Otherwise, claims are not limited to corresponding structure. NVIDIA also argues that corresponding structure is not a sufficiently definite structure. NVIDIA cannot have it both ways. Unlike *MTD Prods.*, ACS does not solely seek to rely on the specification’s disclosure, but also the claims’ language. NVIDIA’s lexicography argument is inapt. NVIDIA itself argues “[l]exicography ... is not required where ... [the] construction relies on the ... meaning of the term in the context of the specification.” Joint Br. at 4-5.

NVIDIA’s discussion of the prosecution history conflates the terms with a “‘module’ term.” But even NVIDIA concedes this was a *different* limitation. The cited discussion is therefore irrelevant. When read in context, these terms would have connoted sufficiently definite structure to a POSITA. No additional construction under § 112, ¶ 6 is necessary.

b. NVIDIA all but concedes its functions are incorrect

NVIDIA’s discussion of function is baseless, and the “reasons discussed above” NVIDIA relies on are conclusory. NVIDIA ignores *BBA Nonwovens*, which addressed “corona means cooperating ... and positioned for electrostatically charging.” 303 F.3d at 1343. The Court rejected the function including “the expression following ... ‘positioned’ [which] describes *where* the corona means is,” not its function. *Id.* at 1344. Here, the phrases after “using” describe *what* the mechanism uses. “A mechanism” in these terms forms the antecedent basis for “the mechanism” in later limitations. The mechanism must perform both functions.

c. NVIDIA includes unnecessary components while ignoring equivalents to propose an unjustifiably narrow structure

For claims 1, 26, 29, and 35, NVIDIA identifies: (1) the MPI module; (2) the RMQ and MRQ “configured to perform the process described at 25:29–43.” For claim 26, NVIDIA adds (3) the “mpiSend, mpiRecv, and mpiTest commands described at 14:1–28” to the MPI module. NVIDIA’s structures are improper because NVIDIA misidentifies the functions. *See Generation II Orthotics*, 263 F.3d at 1364. Only attorney argument asserts otherwise. These components are also unnecessary. NVIDIA does not rebut that “[§ 112, ¶ 6] does not permit incorporation of structure ... beyond that *necessary* to perform the claimed function.” *Asyst Techs., Inc. v. Empak, Inc.*, 268 F.3d 1364, 1369–70 (Fed. Cir. 2001); *see also Northrop*, 325 F.3d at 1354.

Both of NVIDIA’s identifications of the MPI module are unnecessary and improper. The cluster node module is a preferred embodiment of the messaging modules, Ex. B, 11:42–45, cl. 4;

Ex. D, ¶ 141, a preferred embodiment of which includes the MPI module. Ex. B, 12:48–49; Ex. D, ¶ 142. NVIDIA wrongly asserts that this preferred embodiment of a preferred embodiment is necessary to perform the functions. *See* Ex. D, ¶¶ 141–42. Neither of these modules, even at the lower API level, are the only messaging modules disclosed. *Id.*, ¶¶ 139 (node modules), 140 (cluster node modules), 142 (MPI module), 144 (advanced functions module); Ex. B, 11:66–12:2 (MathLink toolkit). Thus, MPI module 302 is not necessary to perform the recited function.

Both of NVIDIA’s identifications of the MRQ and RMQ are unnecessary and improper. For claims 1, 29, and 35, NVIDIA does not rebut ACS’s expert’s conclusion that the MRQ and RMQ are *unnecessary* to perform peer-to-peer communication. Ex. D, ¶¶ 147, 157; *see* Ex. E, ¶¶ 131–35. NVIDIA’s hyper-specific identification of “25:29–43” highlights the unnecessary nature of the MRQ and RMQ to non-asynchronous calls, Ex. B, 25:31–34 (referencing `mpiIrecv()`). The “Basic MPI Calls” overview does not discuss the MRQ or RMQ like the “Asynchronous MPI Calls” overview does. *Compare id.*, 13:41–14:43 *with id.*, 13:9–40. NVIDIA misreads “[t]his process” and “[v]ia this mechanism,” which refer to the cluster node module’s operation. *Id.*, 24:35 (titled “Cluster Node Module Operation.”). NVIDIA’s citation to the concededly “general[]” discussion does not prove the MRQ and RMQ’s necessity, only their potential uses. Likewise, the specific “configur[ation] to perform the process described at 25:29–43” is unnecessary. Even for claim 26, NVIDIA disregards that the MRQ and RMQ are a preferred embodiment of the cluster node module preferred embodiment. Ex. D, ¶¶ 146, 173–74; Ex. B, 21:63–64, 22:17–18. The specific configuration of the MRQ and RMQ is also unnecessary for claim 26. For claim 26, NVIDIA’s identification of the “`mpiSend`, `mpiIRecv`, and `mpiITest` commands described at 14:1–28,” is unnecessary. These calls are within unnecessary modules.

Ex. B, 12:48–49. Call-level identification is also unnecessary; NVIDIA does not identify basic MPI *calls* for claims 1, 29, and 35. *See id.*, 13:25–40. It is also unnecessary for claim 26.

NVIDIA’s arguments against ACS’s proposed corresponding structures fail. NVIDIA complains that the specification does not use “messaging modules.” NVIDIA ignores that means-plus-function claims “shall be construed to cover the corresponding structure ... described in the specification ***and equivalents thereof.***” *Asyst Techs.*, 268 F.3d at 1368 (quoting 35 U.S.C. § 112, ¶ 6). NVIDIA disregards the specification’s disclosure of multiple messaging modules. *See, e.g.*, Ex. D, ¶ 139 (citing Ex. B, 3:46–49)), 140, 142, 144; Ex. B, 11:66–12:2. These examples and their equivalents would inform a POSITA that the corresponding structures are ACS’s structures.

NVIDIA’s conclusory assertion that such structure would be “too generic to constitute structure” fails because a POSITA would have understood the connoted structure to be a specific class. Ex. D, ¶ 145; *Vistan Corp. v. Fadei USA, Inc.*, 547 F. App’x 986, 989–90 (Fed. Cir. 2013) (unpublished) (finding “mechanical linear actuators” were “a distinct and identifiable class of actuators.”). ACS properly relies on expert testimony to prove this. *Vistan Corp.*, 547 F. App’x at 990. NVIDIA’s cases do not prove otherwise. *Med. Instrumentation* is inapposite and rejected “software” because of “a failure to associate software with the converting function.” 344 F.3d at 1211. Correspondence was the issue, not generality. *Id.* at 1210–12. ACS’s structures correspond to the identified functions. Ex. D, ¶¶ 148, 156, 175. *Synchronoss Techs.* is distinguishable because its structure would have “correspond[ed] to ***every known way*** of achieving the [] function.” 987 F.3d at 1367. *Triton Tech* similarly found that “[d]isclosing the broad class of ‘numerical integration’ ... is hardly more than a ***restatement of the integrating function.***” 753 F.3d at 1379. ACS’s structures limit the identified messaging modules to those that are (a) peer-to-peer, and (b) use message-based communication, and for claim 26, (c) are asynchronous. NVIDIA ignores the

components that add specificity. NVIDIA mischaracterizes ACS's expert's statement. Although a POSITA's understanding of the specification's disclosure would be broader than the preferred embodiments, it is not as broad as that in *Triton Tech*, which was not limited beyond the function itself. The Court should adopt ACS's structures.

4. Defendants' Sur-Reply Position

While ACS concedes the claimed "mechanism" does not "connote sufficiently definite structure 'without more'" (Joint Br. at 58), the "more" on which ACS relies cannot supply the requisite structure. ACS first asserts that "peer-to-peer architecture" and "asynchronous calls" provide the requisite structure. *Id.* at 58. This argument ignores the fact that claim 29 lacks either supposed structure. Even for other independent claims, these features are not *part of* the claimed "mechanism," but rather are *used by* mechanism. Joint Br. at 58–59. ACS itself recognizes this distinction when it says: "the phrases after 'using' describe *what* the mechanism uses." Joint Br. at 59 (emphasis in original). ACS' reliance on *Dyfan* is also inapt because it found that "'code' and 'application' (which themselves connote structure)" when combined with specific claim features regarding "the code or application's operation would have connoted structure." *Dyfan, LLC v. Target Corp.*, 28 F.4th 1360, 1368–69 (Fed. Cir. 2022). Here, the claim uses the generic "mechanism" and the two cited features do not connote sufficient structure. For example, a peer-to-peer architecture "is a generalized methodology of intercommunication between cluster nodes, rather than a structure for implementing that methodology." Foster Decl., ¶¶ 123–125.

ACS next asserts that the claims allegedly "describe inputs, outputs, and how outputs are sent" and that this somehow connotes the structure of the claimed "mechanism." Joint Br. at 58. However, ACS never actually identifies what structure this allegedly connotes. *Egenera, Inc. v. Cisco Sys., Inc.*, 972 F.3d 1367, 1374–75 (Fed. Cir. 2020) (rejecting "inputs, outputs, connection

and operation” argument as insufficient to avoid § 112 ¶ 6 “black-box claiming”). ACS’ reliance on *Triplay* fails because its generic construction here – “hardware and/or software modules on the nodes that enable communication” – does not connote sufficient structure. *Triplay, Inc. v. Whatsapp, Inc.*, 13-cv-1703, 2016 WL 3574012, at *7–9 (D. Del. Jun. 30, 2016).

ACS raises three other arguments on the sufficiency of structure. ACS cites the ’768 patent at 14:35–43 (Joint Br. at 58), but the “collective calls” it cites are part of the MPI module in NVIDIA’s structure. *Id.* at 13. ACS also argues that “mechanism” represents a “class of known structures” but does not identify the class. Joint Br. at 58. ACS waived this argument (LR 7.1.3(c)(2)), which fails in any event because mechanism is a nonce term and the addition of “peer-to-peer architecture” and “asynchronous calls” does not impart sufficient structure. Foster Decl., ¶¶ 122–125. Finally, ACS asserts that claim 3 “confirms” the structure in the independent claims (Joint Br. at 59), but fails to explain what structure the “message passing interface” of claim 3 connotes or whether it could perform the claimed function. Joint Br. at 51–52.

ACS also incorrectly asserts that “NVIDIA’s structures are improper because NVIDIA misidentifies the functions.” Joint Br. at 60. However, the only function present in ACS’ proposed construction that is not expressly included in NVIDIA’s proposed function is communicating user instructions, and the specification discloses that user instructions are communicated by the MPI structures identified by NVIDIA. ’768 at 25:38–46, 12:48–53, 13:10–14.

With respect to the parties’ identified structures, ACS’ structure simply replaces one black box (“mechanism”) with another (“messaging modules”). Joint Br. at 55–56. Unlike the “linear actuators” in *Vistan*, “messaging modules” have no accepted meaning. Foster Decl., ¶ 137; *Vistan Corp. v. Fadei USA, Inc.*, 547 F. App’x 986, 989–90 (Fed. Cir. 2013). In fact, ACS’ generic

structure merely reinforces “that the term should be construed as a means-plus-function claim.” *Alarm.com, Inc. v. SecureNet Techs., LLC*, 2019 WL 3996883, at *6 (D. Del. July 23, 2019).

ACS’ criticisms of NVIDIA’s proposed structure should also be rejected. First, ACS’ assertion that NVIDIA’s structures are unnecessary to perform the claimed functions is meritless because the MPI module, RMQ, and MRQ are *expressly* identified as necessary parts of “this mechanism” and “this process [that] provides peer-to-peer behavior” for all communications in the specification. ’768 at 12:41–49, 13:9–16, 16:40–64, 25:38–39; Foster Decl., ¶¶ 128–133; Joint Br. at 55. Second, what ACS cites as other “messaging modules” cannot be the claimed corresponding structure because they (1) are not identified by the specification as being the mechanism and (2) communicate using the corresponding structures identified by NVIDIA. Joint Br. at 61; Foster Decl., ¶¶ 128–130; ’768 at 12:41–49, 13:9–16, 16:40–64, 25:38–39. Third, ACS’ objection to the queues as “a preferred embodiment of the cluster node module” is contrary to law which requires identification of specific structure, including preferred embodiments. Joint Br. at 61; *Signtech USA, Ltd. v. Vutek, Inc.*, 174 F.3d 1352, 1356 (Fed. Cir. 1999). Moreover, the asynchronous commands NVIDIA identifies as additional claim 26 structure are correct as they are necessary for asynchronous calls. ’768 at 13:41–14:30; Foster Decl., ¶ 134.

H. “First Node” Term (claim 35)

communicate a result of the second mathematical expression evaluation to the first node / “the first node”	
ACS	communicate a result of the second mathematical expression evaluation to the <u><i>second</i></u> node
NVIDIA	Indefinite; lacks antecedent basis

1. Plaintiff's Opening Position

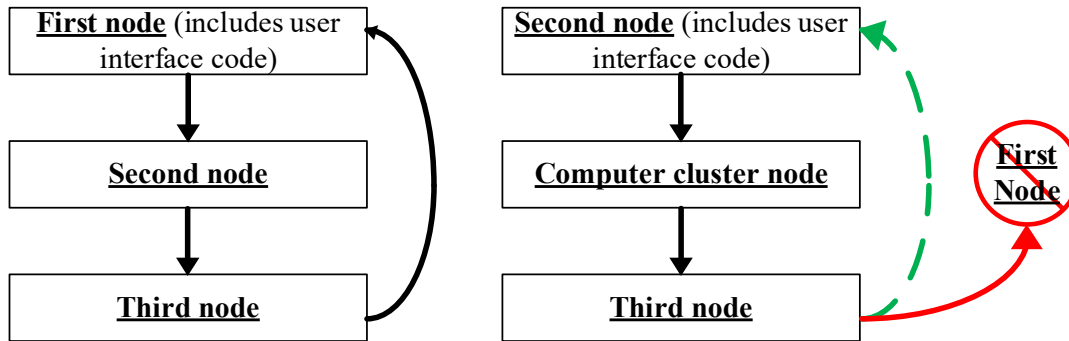
A POSITA would have readily understood that “the first node” in the “First Node” Term was an Examiner’s error, and should instead read “the second node.” Ex. D, ¶ 177. “[A] district court can construe a patent to correct an obvious error ... if (1) the correction is not subject to reasonable debate based on consideration of the claim language and the specification and (2) the prosecution history does not suggest a different interpretation of the claims.” *Bayer Intell. Prop. GmbH v. Taro Pharm. Indus. Ltd.*, 2019 WL 1466193, at *3 (D. Del. Apr. 2, 2019) (citing *Novo Indus., L.P. v. Micro Molds Corp.*, 350 F.3d 1348, 1355 (Fed. Cir. 2003)).

a. “The first node” is an obvious error.

“The first node” in Claim 35 plainly lacks antecedent basis because there is no previous recitation of “a first node.” Ex. B, Cl. 35; Ex. D, ¶ 178. This error is clear from the claim. Ex. D, ¶ 178. Both NVIDIA and its IPR expert conceded that “[t]he term ‘the first node’ lacks antecedent basis and likely is a drafting error.” Ex. U at 63; Ex. G at 80; Ex. D, ¶ 179; *see also* Ex. E, ¶ 71.

b. The “second node” correction is not subject to reasonable debate and is supported by the prosecution history.

The claims and specification support correcting “the first node” to “the second node.” Claims 1, 26, and 29 recite a common communication pattern where the result of a second mathematical expression evaluation is returned to the user interface node. Ex. D, ¶¶ 181–82, 185; *compare* Ex. B, Cls. 1, 26, 29 *with* Ex. B, Cl. 35. The specification also supports this correction by emphasizing the return of results to the user interface node. Ex. B, 6:22–25 (“Results of evaluations ... are communicated back to the first cluster node module ... which communicates them to the user interface module 208.”), 3:12–26, 23:3–5. In Claim 35, the “first node” is not connected to the user interface, but the second node is. Replacing “the first node” with “the second node” restores the common communication pattern:



Ex. D, ¶¶ 182, 190–91.

The prosecution history confirms the “second node” correction. Ex. D, ¶¶ 183–91. Immediately before the application was granted, the Examiner proposed rewriting Claim 35 “to have all the limitations of th[e] other independent claims.” *Id.*, ¶ 183; Ex. C at 103 (Interview Summary). Thus, the Examiner proposed transposing the common communication pattern from Claims 1, 26, and 29 to Claim 35. Ex. C at 103 (Interview Summary); *see also id.* at 63, 65 (Interview Summary), 81–82 (Office Action Response at 13–14); Ex. D, ¶ 183. But because Claim 35 claimed a node instead of a cluster, this transposition did not fit. Ex. D, ¶¶ 184–85, 189–90. The Examiner introduced the antecedent basis error and broke the intended common communication pattern. *Id.* NVIDIA concedes this. Ex. I at 19–20 (“The applicant did not originally amend [Claim 35] ... the examiner amended [Claim 35] ... These amendments ... require specific nodes to perform the specific 1-2-3-1 order”).

The claim language and the specification leave no room for *reasonable* debate: the result of the second mathematical expression evaluation must be returned to the user interface node, for Claim 35, the second node. *Id.*, ¶¶ 180, 183, 192. The prosecution history does not suggest otherwise; it clearly explains the error. The Court should correct the Examiner’s error.

2. Defendants' Answering Position

Claim 35 recites “[a] computer cluster node,” “a second node,” and “a third node,” and then also recites “*the* first node” without having previously mentioned “a first node.” The lack of antecedent basis for the term “the first node” renders the claim indefinite, because it “fail[s] to inform, with reasonable certainty, those skilled in the art about the scope of the invention.” *Nautilus, Inc. v. Biosig Instruments, Inc.*, 572 U.S. 898, 901 (2014).

A POSITA would not know with reasonable certainty whether the term “the first node” refers back to the first node recited in the claim (the “computer cluster node”), an additional node other than those previously recited, or some other node. Construing “the first node” to refer to the computer cluster node would make sense linguistically because the computer cluster node is the first node recited in the claim and would be reasonable functionally because doing so would return a result back to the previous node, allowing for further evaluation. Foster Decl., ¶¶ 74–75. However, construing “the first node” to refer to an additional node not previously recited in the claim would also be reasonable. Foster Decl., ¶¶ 76–78. Nothing in the patent limits the total number of nodes to only three. To the contrary, Figure 2 expressly depicts five nodes, and the written description teaches that “[c]ommunications can occur between any two or more cluster node modules.” ’768 at 6:9–10.

ACS insists that the Court rewrite the claim term “the first node” to read “the second node.” ACS’ argument only further highlights the term’s indefiniteness, presenting a third option for a POSITA to guess at. Moreover, as noted by ACS, claim language can be rewritten only “if (1) the correction is not subject to reasonable debate based on consideration of the claim language and the specification and (2) the prosecution history does not suggest a different interpretation of the claims.” Joint Br. at 66 (citations and quotations omitted). Neither requirement is met here.

For requirement (1), ACS argues that the other independent claims “recite a common communication pattern where the result of a second mathematical expression evaluation is returned to the user interface node.” Joint Br. at 66. ACS’ argument elides a critical distinction between claim 35 and the other independent claims. Claim 35 does not require that the “result of the second mathematical expression evaluation” be returned to a user interface, but instead recites that “the user connection interface is configured to return *at least one result of mathematical expression evaluation* to a user interface or a script.” In contrast, the other independent claims do require that the result of the second mathematical expression evaluation itself be returned to the user interface, *e.g.*, in claim 1: “wherein the first node is configured to return *the result of the second mathematical expression evaluation* to the user interface.” ACS also points to generic statements in the specification that results can be returned to the user interface (Joint Br. at 66), but none of them requires that the specific result at issue—the result of the second mathematical expression evaluation—be returned to the user interface. Accordingly, nothing in the claims or specification resolves the issue past reasonable debate, and correction is not allowed. *Novo Industries, L.P. v. Micro Molds Corp.*, 350 F.3d 1348, 1357–55 (Fed. Cir. 2003) (“Under these circumstances, in order to make sense out of the patent, the district court was required to guess as to what was intended. That is beyond its authority.”).

For requirement (2), ACS cites a line from an Interview Summary proposing an amendment to claim 35 “to have all the limitations of that [sic] other independent claims.” Joint Br. at 67 (quoting Ex. C, at 103). ACS’ argument fails for several reasons. First, the Federal Circuit holds that courts do “not have authority to correct” patents unless the error is “evident on the face of the patent,” even if an error would have been clear from the prosecution history. *Group One v. Hallmark Cards*, 407 F.3d 1297, 1302–03 (Fed. Cir. 2005). As established above, the error

here is not evident on the face of the patent. Second, the Examiner’s statement is far from clear. The Examiner’s statement does not discuss any “common communication pattern,” as ACS contends, and even if ACS’ proposed “correction” were made, claim 35 would not have “all the limitations” of the other independent claims. For example, claim 35 does not require the result of the second mathematical expression evaluation to be passed to the user interface. Finally, the interrogatory responses cited by ACS do not justify rewriting claim 35 but were instead cited out of context from a discussion on prosecution history estoppel.

Accordingly, the Court should reject ACS’ request to rewrite “the first node” to read “the second node” and hold that the claim is indefinite.

3. Plaintiff’s Reply Position

NVIDIA does not dispute that there is a clear error, but rather attempts to prevent the Court from correcting the clear error. There is no reasonable debate on the proper correction. NVIDIA’s alternatives conflate possibility with reasonableness. A POSITA would not have understood the first node to be (1) the computer cluster node simply because it “would make sense linguistically” and be “reasonably functional;” or (2) an unidentified new node simply because “[n]othing in the patent limits the total number of nodes to only three.” The correction is not subject to *reasonable* debate based on the claim language and the specification. *See* Ex. D, ¶¶ 180, 192.

Claims 1, 26, and 29 cover part of the common communication pattern in the following limitations: (1) “a first node comprising ... a user interface;,” and (2) a “third node is configured to ... communicate the result of the second ... evaluation to the first node [*i.e.*, the node comprising the user interface]”). Ex. B, 30:37–59, 32:47–67, 33:55–34:8. ACS proposes a correction to the corresponding limitations in Claim 35: (1) “a second node comprising ... a user interface” and, (2) a “third node is configured to ... communicate a result of the second ... evaluation to the [second

node],” *id.*, 35:28–30, 35:40–36:4, because the second node comprises the user interface. This is the relevant limitation added by the Examiner that has the antecedent basis error.

NVIDIA attempts to confuse the analysis by referencing separate limitations regarding how the nodes thereafter return results to the user interface. *See id.*, 36:5–7. In claims 1, 26, and 29 “the first node is configured to return the result of the second ... evaluation to the user interface.” *Id.*, 30:60–62, 33:1–3, 34:10–12. Claim 35 does not have a corresponding limitation that addresses the second node returning the result of the second evaluation to the user interface. This does not constitute a “critical,” or even relevant, distinction.

NVIDIA also misinterprets its authority on when it is proper to consult the prosecution history. The *error* must be evident from the face of the patent, *Group One*, 407 F.3d at 1303, but the prosecution history may be consulted to determine the proper correction, *see Hoffer v. Microsoft Corp.*, 405 F.3d 1326, 1331 (Fed. Cir. 2005) (“the correct antecedent claim is apparent from the prosecution history”). The Examiner’s stated intent to make claim 35 have “all the limitations of th[e] other independent claims” accompanying the Examiner’s amendment is clear. Ex. D, ¶¶ 183–91; Ex. C at 103 (Interview Summary). That the Examiner did not ultimately transpose literally *all* of the limitations of the other independent claims to claim 35 does not make his intent less clear with respect to the limitation at issue. Tellingly, NVIDIA’s expert does not opine that the prosecution history suggests a meaning other than ACS’s correction. *See* Ex. E, ¶¶ 84–86. More tellingly, NVIDIA independently came to the same conclusion as to the proper correction. *See* Ex. I at 19–20 (“the examiner amended [claim 35] ... [t]hese amendments ... require specific nodes to perform the specific 1-2-3-1 order”). NVIDIA does not substantively address its admission and simply quibbles that its analysis was cited out of context.

4. Defendants' Sur-Reply Position

ACS attempts to improperly shift the burden to NVIDIA to “prevent the court from correcting the error.” But as the party seeking the correction, the burden is on ACS to prove claim 35 can be corrected. In light of the intrinsic evidence, there are at least three different reasonable proposed interpretations for the first node term—thus precluding ACS’ request for judicial correction. This case is thus much different from *Hoffer*, in which the Court allowed correction of an “obvious administrative error” in claim numbering. *Hoffer v. Microsoft*, 405 F.3d 1326, 1331 (Fed. Cir. 2005). Here, claim 35 does not itself resolve the ambiguity. Although claim 35’s “second node” is configured to access code for a user interface, nothing in claim 35 requires “returning the result of the second evaluation to the user interface,” as ACS concedes. Joint Br. at 71. Likewise, ACS’ effort to divine an “intent” from the Examiner’s statement falls far short, with ACS admitting that the amended claim would not include “all the limitations” from the other claims even with ACS’ supposed “correction.” *Id.* Finally, ACS argues that NVIDIA’s expert agrees with ACS, but Dr. Foster makes clear that the claim scope is unclear to a POSITA. Foster Decl., ¶¶ 74–82. Accordingly, judicial correction is not allowed here.

I. “Configured to Access ... Comprising ...” Terms (claims 1, 26, 29, 35)

ACS	“plain and ordinary meaning”
NVIDIA	“each of the nodes is programmed to access a non-transitory computer-readable medium storing program code for a single-node kernel” (claims 1, 26, 29) “a hardware processor programmed to access one or more non-transitory memory devices storing program code for a single-node kernel” (claim 35)

1. Plaintiff’s Opening Position

This term was added after Plaintiff served its opening claim construction brief.

2. Defendants' Answering Position

Despite the Court's order (D.I. 279) ACS contends that this term has a "plain and ordinary meaning" but will not say what it is.⁹ However, ACS' statements to the Court in a discovery dispute and correspondence to NVIDIA reveals that ACS' interpretation of "plain and ordinary meaning" is manifestly incorrect. ACS asserts that this term is "not itself a limitation of the claimed invention" and "the Accused Products do not require software loaded at the time of sale to infringe." D.I. 263, at 2; *id.*, Ex. 17, at 2.

As established above in relation to the "third node" claim term, "configured to" requires that the claimed apparatus actually be configured to operate as specified by the claim; mere capability is not enough. Moreover, ACS' extreme position—reading out the term entirely as "not itself a limitation of the claimed invention" and asserting that an accused device does not need to be loaded with program code that practices the claims to infringe—would misstate the scope of even a claim term drawn to capability, which these terms are not. *INVT*, 46 F.4th at 1374 (even a capability-type claim requires "a device with the capability of performing the recited functions when in operation *without any modification or further programming*") (emphasis added). Here, the plain meaning of "configured to" is "programmed to," consistent with the intrinsic evidence and case law. *See* § I.D, *supra*; *Aspex Eyewear*, 672 F.3d at 1349; *Acuity Brands*, 2021 WL 3187439, at *7.

The use of the term "comprising" in the claim limitation also does not allow the limitation to be ignored. Instead, "comprising" likewise requires the non-transitory computer-readable medium to actually contain the recited program code—which NVIDIA's construction ("storing")

⁹ NVIDIA is providing the first argument on this term, because ACS did not provide argument about the term in its opening brief.

makes clear. *Genentech, Inc. v. Chiron Corp.*, 112 F.3d 495, 501 (Fed. Cir. 1997) (“‘Comprising’ is a term of art used in claim language which means that the named elements are essential.”); *Mars, Inc. v. H.J. Heinz Co., L.P.*, 377 F.3d 1369, 1376 (Fed. Cir. 2004) (citing MPEP for the proposition that “comprising” “is synonymous with ‘including,’ ‘containing,’ or ‘characterized by’”). This claim requirement is consistent with the specification, which repeatedly teaches that the “kernel resides in the at least one computer-readable medium.” ’768 at 2:30–57. The specification uses “resides in” as a synonym for “stored”: “The kernel modules 206a-e can reside in the memory of one or more computer systems on which they run. For example, the memory 114 of the first computer system 110 can store instances of kernel modules 206a-b.” *Id.* at 11:23–26. Thus, “comprising” means “storing” or “containing” program code for a single-node kernel.

The prosecution history is in accord. The examiner repeatedly rejected the claims for double patenting over a parent patent (U.S. Patent No. 8,676,877). Ex. C, at 7–8, 45–46; Ex. 1 at 4–5. In each instance, the examiner equated “configured to” in the ’768 patent with “stored in” in the ’877 patent. Ex. C, at 7–8, 45–46; Ex. 1 at 4–5.

Accordingly, the Court should adopt NVIDIA’s constructions.

3. Plaintiff’s Reply Position

The claims should be given their plain and ordinary meaning. NVIDIA rewrites “configured to” as “programmed to,” and “comprising” as “storing.” NVIDIA never explains why these non-technical words require construction. If helpful, the ordinary meaning of “configured to” can be articulated as “designed to.” Using “designed to” in place of “configured to” is consistent with the specification. *Compare* Ex. B, Abstract (“a cluster node module is configured to communicate with the kernel”), 3:18–21, 2:51–54, 5:64–6:2 *with* Ex. B, 1:37–41 (“a kernel that is designed to communicate with a [] node.”), 2:18–19, 4:58–59. The specification never uses

“programmed to.” NVIDIA itself cites a case construing “configured to” as “designed to.” Joint Br. at 21 (citing approvingly *Acuity Brands*, 2021 WL 3187439, at *7); *Aspex Eyewear*, 672 F.3d at 1349 (equating configured to and designed to).

The ordinary meaning of “comprising,” can be further articulated as “including.” *Exergen Corp. v. Wal-Mart Stores, Inc.*, 575 F.3d 1312, 1319 (Fed. Cir. 2009) (“‘comprising,’ ... is well understood ... to mean ‘including but not limited to.’”); *Mars*, 377 F.3d at 1376. This is consistent with the specification’s disclosure that the “kernel resides in the at least one computer-readable medium.” Ex. B at 2:30–57. NVIDIA’s proposed alternative, “storing,” is inconsistent with other claims that expressly recite “stored.” *Id.*, cls. 5, 6, 28, 36. Moreover, NVIDIA’s construction would improperly render claim 5 surplusage. *Id.*, cl. 5 (“single-node kernel is stored in a [] computer-readable medium”).

Further articulating the meaning of “configured to” and “comprising,” however, does not address the real dispute: NVIDIA’s attempt to improperly rewrite the claims to require actual performance of a functional limitation and add the “non-transitory computer-readable medium”/“memory devices” and “program code for a single-node kernel” as independent structural elements. Practitioners and courts have long held that apparatus claims reciting an apparatus “configured to” perform a function do not require accused devices to actually perform the recited function. *See INVT*, 46 F.4th at 1371. However, that is exactly what NVIDIA seeks.

For example, claim 1 recites “each of the nodes is configured to [perform a function].” The object of the claim is the nodes. The language following “configured to” describes a function the nodes must be configured to perform. The computer-readable medium and single-node kernel are part of the function and describe the environment in which the nodes perform their function. *Id.* at 1371, 1375. ACS does not “read out” or “ignore” terms as NVIDIA alleges. ACS simply

recognizes that the claims as written do not recite the computer-readable medium and single-node kernel as independent structural elements. NVIDIA’s attempt to convert them into required structure is contrary to the plain meaning of the limitations and inconsistent with the prosecution history. Where ACS intended to claim the computer-readable medium and kernel as independent structural elements, it did so explicitly. *See, e.g.*, Ex. W (U.S. Patent No. 8,082,289), cl. 1 (“A cluster computer comprising: ... at least one computer-readable medium ... a first kernel”).

To argue otherwise NVIDIA mischaracterizes its cases.¹⁰ The distinction *Typhoon Touch* draws between “‘a memory that must perform the recited function [(i.e., storing data collection applications)]’ and ... ‘... memory [that] is capable of being configured to store data collection applications’” is not based on the “data collection configured to” language, but rather the “memory for” language. *See* 659 F.3d at 1380. NVIDIA also presents a false dichotomy between required performance and theoretical capability. But the cases requiring capability without modification do not mean actual performance of function. *See id.* ACS also does not assert the “capable of” construction rejected by *Aspex Eyewear*. NVIDIA’s discussion of *INVT* fares no better. NVIDIA misleadingly cites to *Ball Aerosol*, which was **distinguished** by *INVT* as having “little relevance” because of *Ball Aerosol*’s claim language and technological field, mechanical candle holders. *See INVT*, 46 F.4th at 1372–73 (distinguishing *Ball Aerosol*, 555 F.3d at 994–95). NVIDIA also parrots *INVT*’s citation to *Parkervision*, 903 F.3d at 1361, but ignores *INVT*’s holding that “[s]ometimes a device only needs to be ‘capable of operating’ according to a claimed limitation, for ... infringement ... [w]hether infringement requires actual performance ... depends on the

¹⁰ NVIDIA’s cases related to “configured to” that were improperly presented in the earlier “third node” phrase section, *see supra* § II.D.2, are addressed here.

claim language,” *INVT*, 46 F.4th at 1371. NVIDIA’s arguments ignore the claim language, the apparatus-method distinction, and the technological field. *See id.* at 1371–73.

NVIDIA’s belated attempt to create a claim construction dispute is disingenuous and demonstrates that NVIDIA is not seeking a proper claim construction for the disputed terms. As one example, claim 1 recites no less than eight elements that contain the common “configured to” language. The claims use “configured to” over 60 times to introduce functional limitations. This language is ubiquitous in patent claim drafting. ACS has provided three sets of infringement contentions to NVIDIA relying on the plain and ordinary meaning of “configured to” for all “configured to” limitations, not just the one plucked for construction by NVIDIA. Yet, NVIDIA now seeks to construe a single “configured to” phrase because it wants to create a non-infringement argument that is contrary to law. NVIDIA seeks to have this Court construe this phrase to require that each accused product have software capable of performing mathematical evaluations (*i.e.*, the single-node kernel) in memory (*i.e.*, the claimed non-transitory computer-readable medium) at the time of sale. But that is not what the law requires, and NVIDIA’s failure to seek to construe any other “configured to” phrase shows they do not seek a proper claim construction.

Radware does not justify a “programmed to” construction; *Radware* considered whether “configured to” required user-selected settings or only software capability. 2014 WL 1572644, at *12–13. That software distinction is not at issue here. NVIDIA’s cases regarding “comprising” are inapposite. The “comprising” here is not the transitional term at issue in *Genentech*, 112 F.3d at 497, 501 and discussed in *Mars*, 377 F.3d at 1372, 1376, which introduces the claims’ requirements. “Comprising” is encompassed within a function and is not independent structure of the claimed invention. *See INVT*, 46 F.4th at 1375.

NVIDIA's citation's use permissive language that distinguishes "comprising" from "storing" and "residing." Ex. B, 11:23–26 ("kernel[s] [] *can* reside in the memory ... memory ... *can* store ... kernel[s]"); *see also id.*, 11:27–30. NVIDIA also cites a particular embodiment. *See id.*, 2:30–57. This is not definitional. NVIDIA's citation also clarifies that the kernel need not be in the memory of the computer system that runs it. *Id.*, 11:23–25 ("kernel modules [] *can* reside in the memory of ... computer systems on which they run"). This is consistent with functional interpretation of "configured to," which may be met regardless of where the memory is.

NVIDIA misreads the prosecution history. Even if the Examiner equated "configured to" with "stored in" (which he did not), it would not support NVIDIA's construction. To the extent NVIDIA intended to argue that the Examiner equated "comprising" with "stored in," NVIDIA is wrong. There is no claim limitation in the '768 Patent that recites "comprising" that corresponds to a limitation in the '877 Patent that recites "stored in."

The Court should give these terms their plain and ordinary meaning, if helpful, by articulating "configured to" as "designed to" and "comprising" as "including."

4. Defendants' Sur-Reply Position

As this Court has recognized, the Federal Circuit holds that "'configured to' has a narrower definition than 'having the capacity to' or 'capable of.'" *Acuity Brands Lighting, Inc. v. Ultravision Techs, LLC*, 19-2207-MN, 2021 WL 3187439, at *7 (D. Del. 2021) (quoting *Aspex Eyewear, Inc. v. Marchon Eyewear, Inc.*, 672 F.3d 1335, 1349 (Fed. Cir. 2012)). The Federal Circuit recently described "configuration-type" claims as "requir[ing] an infringing device to actually perform and operate according to the functional terms recited in the claim" when put into operation. *INVT SPE LLC v. Int'l Trade Comm'n*, 46 F.4th 1361, 1371–72 (Fed. Cir. 2022).

Here, the claims expressly recite “configured to” limitations, not “capable of” limitations. Accordingly, this case is like *Nevro*, where the Federal Circuit rejected an attempt to use “designed to” to cover mere capability, instead construing the term “configured to” to mean “programmed to.” *Nevro Corp. v. Boston Sci. Corp.*, 955 F.3d 35, 41 (Fed. Cir. 2020). ACS fails to explain how “designed to” can mean anything other than “programmed to” for these claims. It notes this Court’s decision in *Acuity*, but that decision supports NVIDIA. In *Acuity*, the Court rejected an argument that “configured to” should be construed as “capable of,” and relied on decisions construing “‘configured to’ as ‘programmed to.’” *Acuity*, 2021 WL 3187439, at *7. Moreover, the claims in *Acuity* recited hardware, making “designed to” a better fit. *Id.* In any event, regardless of what words are used to describe “configured to,” the programs needed to perform the recited functions must actually be present, without modification, for infringement to be found.¹¹

ACS also argues that this limitation does not recite any “independent structural elements.” Joint Br. at 75–76. To the extent ACS is arguing that hardware alone is sufficient, it misrepresents the patent, which requires both hardware and software. *E.g.*, ‘768 at claim 1 (reciting both hardware (*e.g.*, “a hardware processor”) and software (“program code”)). *Nazomi Comm’ns, Inc. v. Nokia Corp.*, 739 F.3d 1339, 1345 (Fed. Cir. 2014) (“[T]he claims do not cover hardware that contemplates an environment where it could be combined with software, but rather require a hardware-software combination that must perform the described functions.”). To the extent ACS is arguing the ability of an accused product to be modified with other software would be sufficient for infringement, the Federal Circuit has rejected that argument, even for “capable of” claims.

¹¹ See also *TQ Delta LLC v. Adtran, Inc.*, 14-cv-954-RGA, 2021 WL 1200595, at *4–5 (D. Del. Mar. 30, 2021) (construing “configured to” as “includes the necessary hardware and software for performing the functionality recited in the claim without the need to rebuild, rewrite or recompile the code for, or redesign any of that hardware or software”).

INVT, 46 F.4th at 1374 (an accused product must be “capabl[e] of performing the recited function when in operation without any modification”). And to the extent that ACS is arguing that claimed non-transitory computer-readable medium could hypothetically be in a device other than the accused product, that is no justification to read out the limitation entirely. Instead, ACS would still need to show the other device meets the claim limitations. *See id.* at 1375 (where the accused device’s “capability is dependent on the base station’s capability,” the base station’s “operations must be known to determine whether the accused device infringes”).

Moreover, the Court should construe “comprising” as “storing” rather than “including.” “Storing” removes any ambiguity that the program code must be on the accused products. Contrary to ACS, “storing” is more consistent with the specification, which teaches that the kernel code resides in or is stored in memory. ’768 at 2:30–57, 11:23–26. ACS also ignores that the Summary of the Invention states that the “kernel resides in the at least one computer-readable medium” in the claimed cluster. ’768 at 2:30–37. Finally, “storing” does not render dependent claims superfluous. Claim 5 recites additional limitations that “each” kernel is stored and “configured to accept a request.” ’768 at 31:12–14. Claim 6 adds limitations to claim 5; claims 28 and 36 are not directed to storage of the kernel at all.

Respectfully submitted,

/s/ Emily S. DiBenedetto

Karen E. Keller (No. 4489)
Nathan R. Hoeschen (No. 6232)
Emily S. DiBenedetto (No. 6779)
SHAW KELLER LLP
1105 North Market Street, 12th Floor
Wilmington, DE 19801
(302) 298-0700
kkeller@shawkeller.com
nhoeschen@shawkeller.com
edibenedetto@shawkeller.com
*Attorneys for Plaintiffs Advanced Cluster
Systems, Inc.*

OF COUNSEL:

Jon W. Gurka
Brian Claassen
Cheryl Burgess
Pushkal Mishra
KNOBBE, MARTENS, OLSON
& BEAR, LLP
2040 Main Street, 14th Floor
Irvine, CA 92614
(949) 760 0404

Karl W. Kowallis
KNOBBE, MARTENS, OLSON
& BEAR, LLP
115 Avenue of the Americas, 24th Floor
New York, NY 10036
(212) 849-3024

Ben K. Shiroma
KNOBBE, MARTENS, OLSON
& BEAR, LLP
1925 Century Park East, Suite 600
Los Angeles, CA 90067
(310) 551-3450

Dated: December 21, 2022

/s/ Brian A. Biggs

Brian A. Biggs (DE Bar No. 5591)
Erin E. Larson (DE Bar No. 6616)
DLA PIPER LLP (US)
1201 North Market Street, Suite 2100
Wilmington, DE 19801-1147
302.468.5700
brian.biggs@dlapiper.com
erin.larson@dlapiper.com
*Attorneys for Defendants NVIDIA Corporation,
NVIDIA Singapore Pte. Ltd., and NVIDIA
International, Inc.*

OF COUNSEL:

Mark Fowler
Clayton Thompson
Jake Zolotorev
Carrie Williamson
Monica De Lazzari
DLA PIPER LLP
2000 University Avenue
East Palo Alto, CA 94303-2214
(650) 833-2000

Peter F. Nelson
DLA PIPER LLP
500 8th St, NW
Washington, DC 20004
(202) 799-4092

CERTIFICATE OF SERVICE

I, Emily S. DiBenedetto, hereby certify that on December 21, 2022, this document was served on nvidia-acs-dla@us.dlapiper.com and the persons listed below in the manner indicated:

BY EMAIL

Brian A. Biggs
Stephanie E. O'Byrne
Erin E. Larson
DLA PIPER LLP
1201 North Market Street, Suite 2100
Wilmington, DE 19801-1147
(302) 468-5700
brian.biggs@us.dlapiper.com
stephanie.obyrne@us.dlapiper.com
erin.larson@us.dlapiper.com

David Yang
HAWKINSON YANG LLP
570 Wilshire Boulevard, Suite 1800
Log Angeles, California 90036
(213) 634-0341
dyang@hycounsel.com

Peter F. Nelson
DLA PIPER LLP
500 8th St. NW
Washington, DC 20004
(202) 799-4000
peter.nelson@us.dlapiper.com

Mark Fowler
Clayton Thompson
Jake Zolotorev
Carrie Williamson
Monica De Lazzari
DLA PIPER LLP
2000 University Avenue
East Palo Alto, CA 94303-2214
(650) 833-2000
mark.fowler@us.dlapiper.com
clayton.thompson@us.dlapiper.com
jake.zolotorev@us.dlapiper.com
carrie.williamson@us.dlapiper.com
monica.delazzari@us.dlapiper.com

Matthew Hawkinson
HAWKINSON YANG LLP
5670 Wilshire Boulevard, Suite 1800
Log Angeles, California 90036
(213) 634-0369
mhawkinson@hycounsel.com

/s/ Emily S. DiBenedetto

John W. Shaw (No. 3362)
Karen E. Keller (No. 4489)
Nathan R. Hoeschen (No. 6232)
Emily S. DiBenedetto (No. 6779)
SHAW KELLER LLP
I.M. Pei Building
1105 North Market Street, 12th Floor
Wilmington, DE 19801
(302) 298-0700
jshaw@shawkeller.com
kkeller@shawkeller.com
nhoeschen@shawkeller.com
Attorneys for Plaintiff

EXHIBIT A

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

ADVANCED CLUSTER SYSTEMS, INC.,)	
)	
Plaintiff,)	
)	
v.)	C.A. No. 19-2032-MN-CJB
)	
NVIDIA CORPORATION, NVIDIA)	
SINGAPORE PTE. LTD., and NVIDIA)	
INTERNATIONAL, INC.,)	
)	
Defendants.)	

AMENDED JOINT CLAIM CONSTRUCTION CHART

Pursuant to the Court’s Scheduling Order (D.I. 17), as amended on July 9, 2021 (D.I. 82) and January 10, 2022 (D.I. 107), and the Court’s October 4, 2022 order (D.I. 247), Plaintiff Advanced Cluster Systems, Inc. and Defendants NVIDIA Corporation, NVIDIA Singapore Pte. Ltd., and NVIDIA International, Inc. (collectively, “NVIDIA”) hereby provide the following Amended Joint Claim Construction Chart for U.S. Patent No. 10,333,768 (the “’768 Patent”).

A. Construction of terms on which the parties agree

The parties have not agreed to any constructions.

B. Parties’ constructions of disputed terms

In accordance with the Scheduling Order, the parties’ proposed claim constructions are provided below. Each party reserves the right to rely on any evidence identified below by the other party.

/s/ Emily S. DiBenedetto

Karen E. Keller (No. 4489)
Nathan R. Hoeschen (No. 6232)
Emily S. DiBenedetto (No. 6779)
SHAW KELLER LLP
1105 North Market Street, 12th Floor
Wilmington, DE 19801
(302) 298-0700
kkeller@shawkeller.com
nhoeschen@shawkeller.com
edibenedetto@shawkeller.com
*Attorneys for Plaintiffs Advanced Cluster
Systems, Inc.*

Jon W. Gurka
Brian C. Claassen
Cheryl Burgess
KNOBBE, MARTENS, OLSON & BEAR, LLP
2040 Main Street, 14th Floor
Irvine, CA 92614
(949) 760-0404

Ben K. Shiroma
KNOBBE, MARTENS, OLSON & BEAR, LLP
1925 Century Park East Suite 600
Los Angeles, CA 90067
(310) 551-3450

Karl W. Kowallis
KNOBBE, MARTENS, OLSON & BEAR, LLP
1155 Avenue of the Americas 24th Floor
New York, NY 10036
(212) 849-3000

Dated: October 12, 2022

/s/ Stephanie E. O'Byrne

Brian A. Biggs (DE Bar No. 5591)
Stephanie E. O'Byrne (DE Bar No. 4446)
Erin E. Larson (DE Bar No. 6616)
DLA PIPER LLP (US)
1201 North Market Street, Suite 2100
Wilmington, DE 19801-1147
(302) 468-5700
brian.biggs@us.dlapiper.com
stephanie.obyrne@us.dlapiper.com
erin.larson@us.dlapiper.com
*Attorneys for Defendants NVIDIA Corporation,
NVIDIA Singapore Pte Ltd. and NVIDIA International*

Mark Fowler (*Pro Hac Vice*)
Clayton Thompson (*Pro Hac Vice*)
Jake Zolotorev (*Pro Hac Vice*)
Carrie Williamson (*Pro Hac Vice*)
Monica de Lazzari (*Pro Hac Vice*)
DLA PIPER LLP (US)
2000 University Avenue
East Palo Alto, CA 94303-2214
mark.fowler@dlapiper.com
clayton.thompson@dlapiper.com
jake.zolotorev@dlapiper.com
carrie.williamson@dlapiper.com
monica.delazzari@dlapiper.com

Joint Claim Construction Chart
Disputed Terms

Claim Term	Claim(s)	ACS's Proposed Construction	ACS's Intrinsic Evidence	NVIDIA's Proposed Construction	NVIDIA's Intrinsic Evidence
peer-to-peer architecture	Claims 1 and 35	architecture in which each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node	<p>U.S. Patent No. 10,333,768 at col. 1, ll. 44-62, col. 3, ll. 39-49, col. 6, ll. 26-62, col. 12, ll. 26-40, col. 23, ll. 13-14, 51-56, col. 24, ll. 39-40, 52-53, col. 25, ll. 1-47, col. 26, l. 65-col. 27, l. 67, claims 1-25, 30-39, FIG. 2.</p> <p>File History for U.S. Patent No. 10,333,768 at 2016-11-16 Miscellaneous Internal Document at 1; 2016-11-16 Applicant Initiated Interview Summary at 2; 2016-11-21 Applicant Arguments/Remarks Made in an Amendment at 8-9; 2016-11-21 Claims at 2; 2017-03-09 Final Rejection at 5, 7; 2017-08- 08 Claims at 2,</p>	an architecture in which each node is configured to communicate with other nodes	'768 patent at 1:5-62, 3:39-55, 6:15-29, 12:26-40, 14:45-65, 17:1-45, 21:25-50, 24:35-25:21, 25:22-47, 26:65-27:67, 30:33-67, FIG. 2.

Claim Term	Claim(s)	ACS's Proposed Construction	ACS's Intrinsic Evidence	NVIDIA's Proposed Construction	NVIDIA's Intrinsic Evidence
			<p>8;</p> <p>2017-08-08 Applicant Arguments/Remarks Made in an Amendment at 10-11;</p> <p>2017-08-08 Miscellaneous Internal Document at 1;</p> <p>2017-08-08 Applicant Initiated Interview Summary at 2;</p> <p>2018-04-13 Non- Final Rejection at 5, 7-8;</p> <p>2018-04-13 Miscellaneous Internal Document at 1;</p> <p>2018-10-05 Applicant Arguments/Remarks made in an Amendment at 13-14</p> <p>2018-10-05 Claims at 2, 10; 2018-11-29 Notice of Allowance and Fees Due at 3, 11.</p> <p>IPR2021-00019 Petition for IPR at 1, 5-6, 18-22, 24, 28-32, 42, 53, 68, 70,</p>		

Claim Term	Claim(s)	ACS's Proposed Construction	ACS's Intrinsic Evidence	NVIDIA's Proposed Construction	NVIDIA's Intrinsic Evidence
			<p>76;</p> <p>IPR2021-00019 Patent Owner's Preliminary Response at 1-16, 20-26, 32, 34-48, 50-51;</p> <p>IPR2021-00019 Denial of Institution of IPR at 10-11, 13, 17-19, 21-22;</p> <p>IPR2021-00020 Petition for IPR at 1, 5-7, 16-24, 29-31, 45, 60, 66, 73, 77;</p> <p>IPR2021-00020 Patent Owner's Preliminary Response at 1-14, 23-30, 33- 47, 49-50;</p> <p>IPR2021-00020 Denial of Institution of IPR at 11, 14, 20-22;</p> <p>IPR2021-00019/IPR2021-00020 Petitioners Reply to Patent Owner's Preliminary Response.</p>		
communicate a result of the second mathematical expression	Claim 35	communicate a result of the second mathematical expression evaluation to the second node	U.S. Patent No. 10,333,768 at col. 3, ll. 39-55, col. 5, l. 56-col. 6, l. 25, col. 22, l. 48-col. 23, l. 7, FIG. 2, claims 1-	Indefinite; lacks antecedent basis.	'768 patent at 6:9–29, 35:9–36:4, FIG. 2, claim 35.

Claim Term	Claim(s)	ACS's Proposed Construction	ACS's Intrinsic Evidence	NVIDIA's Proposed Construction	NVIDIA's Intrinsic Evidence
evaluation to the first node/ "the first node"			<p>39.</p> <p>File History for U.S. Patent No. 10,333,768 at 2017-08-08 Claims at 7-8; 2018-04-13 Non-Final Rejection at 26; 2018-10-05 Applicant Arguments/Remarks Made in an Amendment at 12-14; 2018-10-05 Claims at 10; 2018-11-29 Examiner Initiated Interview Summary; 2018-11-29 Notice of Allowance at 11-12; 2018-11-29 Index of Claims.</p> <p>IPR2021-00019 Petition for IPR at 38, 41-47;</p> <p>IPR2021-00019 Patent Owner's Preliminary Response at 26-29, 44-47;</p> <p>IPR2021-00019 Denial of Institution of IPR at 17-18, 23-30;</p> <p>IPR2021-00020 Petition for IPR at 41-50, 63-64, Declaration of Henry</p>		<p>11/29/2018 Notice of Allowability and Examiner Amendment at 12.</p> <p>11/21/2018 Examiner-Initiated Interview Summary 02/14/2014 Amendment at 10.</p>

Claim Term	Claim(s)	ACS's Proposed Construction	ACS's Intrinsic Evidence	NVIDIA's Proposed Construction	NVIDIA's Intrinsic Evidence
			Tufo at 1-5, 80; IPR2021-00020 Patent Owner's Preliminary Response at 18-21, 43-45; IPR2021-00020 Denial of Institution of IPR at 18, 24-29, 30-31.		
cluster node module	Claims 4, 7-8, 10, 27, and 30-31	a module that establishes intercommunication among nodes in a computer cluster and allows exchanging messages among nodes using a peer-to peer-architecture	U.S. Patent No. 10,333,768 at Abstract, col. 1, ll. 44-62, col. 2, ll. 30-57, col. 2, l. 58-col. 3, l. 8, col. 3, ll. 17-26, col. 3, ll. 28-38, col. 4, ll. 56-62, col. 5, l. 16-col. 6, l. 38, col. 11, ll. 13-50, col. 11, ll. 41-54, col. 11, l. 55-col. 21, l. 61, col. 12, ll. 41-53, col. 16, ll. 39-61, col. 21, l. 62-col. 22, l. 24, col. 22, ll. 25-45, col. 22, l. 48-col. 23, l. 7, col. 23, l. 28-col. 26, l. 39, col. 30, ll. 1- 4, FIG. 2, FIG. 3, FIG. 4, FIG. 5, claims 1, 4-8, 10, 14-17, 27-28, 30-32, 35. U.S. Patent No. 7,768,289 at claims 8-9, 13, 26. File History for U.S.	a module running on each cluster node, configured to communicate messages with the single-node kernel on the same node, and with other cluster node modules	'768 patent at 1:25-43, 2:24-29, 2:34-3:8, 4:13-16, 4:58-63, 5:33-6:25, 6:26-30, 11:21-23, 11:32-40, 11:42-12:2, 12:41-46, 23:10-26, 23:51-62, 24:29-31, 24:39-44, 24:52-58, Figures 2-5

Claim Term	Claim(s)	ACS's Proposed Construction	ACS's Intrinsic Evidence	NVIDIA's Proposed Construction	NVIDIA's Intrinsic Evidence
			<p>Patent No. 10,333,768 at 2014-09-08 Claims at 2; 2016-07-20 Non-Final Rejection at 3-4, 14-16; Miscellaneous Internal Document at 2; 2016-11-21 Claims at 2-6; 2017-03-09 Final Rejection at 4-5, 8-12, 17; 2017-08-08 Claims at 2-7; 2017-08-08 Miscellaneous Internal Document at 2; 2018-04-13 Non-Final Rejection at 4-5, 10-14, 21-24; 2018-10-05 Claims at 3-4, 7-9; Notice of Allowance and Fees Due at 4-5, 8, 10-11.</p> <p>IPR2021-00019 Petition for IPR at 5, 14, 34, 52-70, 75-77;</p> <p>IPR2021-00019 Patent Owner's Preliminary Response at 1-6, 8-16, 29-32, 50-52;</p> <p>IPR2021-00019/IPR2021-00020 Petitioners Reply to Patent Owner's</p>		

Claim Term	Claim(s)	ACS's Proposed Construction	ACS's Intrinsic Evidence	NVIDIA's Proposed Construction	NVIDIA's Intrinsic Evidence
			<p>Preliminary Response at 4-7;</p> <p>IPR2021-00019/IPR2021-00020 Patent Owner's Consolidated Sur-Reply In Support Of Its Preliminary Responses at 4-9;</p> <p>IPR2021-00019 Denial of Institution of IPR at 5-6, 10-11, 22;</p> <p>IPR2021-00020 Petition for IPR at 5, 16-17, 26, 53-56, 58;</p> <p>IPR2021-00020 Patent Owner's Preliminary Response at 3-6, 8-15, 21-22, 49-50;</p> <p>IPR2021-00020 Denial of Institution of IPR at 5-6, 11-12, 20.</p>		
kernel	Claims 1, 26, 29, and 35	plain and ordinary meaning: program code	U.S. Patent No. 10,333,768 at Abstract, col. 1, ll. 29-43, col. 1, ll. 47-63, col. 2, ll. 18-23, col. 2, l. 30-col. 3, l. 26, col. 4, ll. 14-23, col. 4, ll. 56-62, col. 5, l. 15-col. 6,	program code for interpreting high-level code, commands, and/or instructions supplied by a user or a script into low-level code, such as, for example, machine	'768 patent at Abstract, 1:29-62, 1:37-43, 2:18-3:55, 4:14-33, 4:37-62, 5:16-6:38, 7:11-32, 8:29-49, 9:48-67, 11:15-12:2, 12:14-40, 13:25-40 (Table

Claim Term	Claim(s)	ACS's Proposed Construction	ACS's Intrinsic Evidence	NVIDIA's Proposed Construction	NVIDIA's Intrinsic Evidence
			<p>l. 29, col. 7, ll. 28-32, col. 8, ll. 44-49, col. 9, ll. 63-67, col. 11, ll. 13-32, col. 11, ll. 41-45, col. 11, l. 55-col. 12, l. 2, col. 12, ll. 5, col. 12, ll. 26-37, col. 13, ll. 41-47, col. 14, ll. 1-28, col. 22, ll. 25-45, col. 22, l. 48-col. 23, l. 8, col. 23, ll. 8-26, col. 23, l. 28-col. 26, l. 39, col. 29, l. 66-col. 30, l. 1, FIG. 2, Table B, Table C, claims 1-39.</p> <p>File History for U.S. Patent No. 10,333,768 at 2014-09-08 Claims at 2; 2016-07-20 Non-Final Rejection at 4, 7-8, 14-17; 2016-11-16 Miscellaneous Internal Document at 1-2; 2016-11-21 Claims at 2-6; 2017-03-09 Final Rejection at 5-18; 2017-08-08 Claims at 2-7; 2017-08-08 Applicant Arguments/Remarks Made in an Amendment at 10; 2017-08-08 Miscellaneous Internal</p>	language or assembly language	<p>B),13:42-46, 14:1-28, 22:26-23:26, 23:34-37, 23:57-24:34, 24:36-38, 24:52-25:61, 26:4-39, 30:28-43, claims 1, 2, 5, 6, 12, 16, 19, 23-32, 35.</p>

Claim Term	Claim(s)	ACS's Proposed Construction	ACS's Intrinsic Evidence	NVIDIA's Proposed Construction	NVIDIA's Intrinsic Evidence
			<p>Document at 1-2; 2017-08-08 Applicant Initiated Interview Summary at 2; 2018-04-13 Non-Final Rejection at 5, 7, 10-18, 21- 24, 26, 29; 2018-05-22 Miscellaneous Internal Document at 1-2; 2018-05- 22 Applicant Initiated Interview Summary at 1; 2018-05-22 Examiner Initiated Interview Summary at 2; 2018-10-05 Claims at 2-10; 2018-11-29 Notice of Allowance and Fees Due at 3-12.</p> <p>IPR2021-00019 Petition for IPR at 1, 5-6, 16-37, 40-44, 47, 49-56, 58-60, 62-69, 71-77;</p> <p>IPR2021-00019 Patent Owner's Preliminary Response at 1, 10-11, 15, 22, 32, 34-47;</p> <p>IPR2021-00019/IPR2021-00020 Petitioners Reply to Patent Owner's Preliminary Response at</p>		

Claim Term	Claim(s)	ACS's Proposed Construction	ACS's Intrinsic Evidence	NVIDIA's Proposed Construction	NVIDIA's Intrinsic Evidence
			<p>5-10;</p> <p>IPR2021-00019/IPR2021-00020 Patent Owner's Consolidated Sur-Reply In Support Of Its Preliminary Responses at 6-9;</p> <p>IPR2021-00019 Denial of Institution of IPR at 5-6, 12-14, 19-23, 29-30;</p> <p>IPR2021-00020 Petition for IPR at 1, 5, 19-20, 22-26, 28-49, 51-67, 72, 76-77;</p> <p>IPR2021-00020 Patent Owner's Preliminary Response at 1-2, 10-12, 14, 21, 25, 31, 33-46;</p> <p>IPR2021-00020 Denial of Institution of IPR at 5-7, 13-15, 20-24, 28.</p>		
single-node kernel	Claims 1, 26, 29, and 35	<p>plain and ordinary meaning:</p> <p>program code that can run on one node</p>	<p>U.S. Patent No. 10,333,768 at col. 1, ll. 22-28, col. 1, ll. 37-43, col. 1, ll. 44- 62, col. 2, ll. 18-24, col. 2, l. 58-col. 3, l. 2-4, 13-26, col. 4, ll. 14-18, 58-62, col. 3, ll.</p>	a kernel that is designed to communicate with and run on only a single node	<p>'768 patent at 1:22-29, 1:37-43, 2:18-21, 2:58-3:2, 3:13-26, 4:58-62, 29:66-30:1, 30:28-32.</p> <p><i>See also</i> intrinsic evidence for "kernel".</p>

Claim Term	Claim(s)	ACS's Proposed Construction	ACS's Intrinsic Evidence	NVIDIA's Proposed Construction	NVIDIA's Intrinsic Evidence
			<p>13-26, col. 29, l. 66-col. 30, l. 4, claims 1-39.</p> <p>File History for U.S. Patent No. 10,333,768 at 2014-09-08 Claims at 2; 2016-07-20 Non-Final Rejection at 4, 7-8, 14-17; 2016-11-16 Miscellaneous Internal Document at 1-2; 2016-11-21 Claims at 2-6; 2017-03-09 Final Rejection at 5-18; 2017-08-08 Claims at 2-7; 2017-08-08 Applicant Arguments/Remarks Made in an Amendment at 10; 2017-08-08 Miscellaneous Internal Document at 1-2; 2017-08-08 Applicant Initiated Interview Summary at 2; 2018-04-13 Non-Final Rejection at 5, 7, 10-18, 21- 24, 26, 29; 2018-05-22 Miscellaneous Internal Document at 1-2; 2018-05- 22 Applicant Initiated Interview Summary at 1; 2018-05-22 Examiner Initiated</p>		

Claim Term	Claim(s)	ACS's Proposed Construction	ACS's Intrinsic Evidence	NVIDIA's Proposed Construction	NVIDIA's Intrinsic Evidence
			<p>Interview Summary at 2; 2018-10-05 Claims at 2-10; 2018-11-29 Notice of Allowance and Fees Due at 3-12.</p> <p>IPR2021-00019 Petition for IPR at 1, 5-6, 16-37, 40-44, 47, 49-56, 58-60, 62-69, 71-77;</p> <p>IPR2021-00019 Patent Owner's Preliminary Response at 1, 10-11, 15, 22, 32, 34-47;</p> <p>IPR2021-00019/IPR2021-00020 Petitioners Reply to Patent Owner's Preliminary Response at 5-10;</p> <p>IPR2021-00019/IPR2021-00020 Patent Owner's Consolidated Sur-Reply In Support Of Its Preliminary Responses at 6-9;</p> <p>IPR2021-00019 Denial of Institution of IPR at 5-6, 12-14, 19-23, 29-30;</p>		

Claim Term	Claim(s)	ACS's Proposed Construction	ACS's Intrinsic Evidence	NVIDIA's Proposed Construction	NVIDIA's Intrinsic Evidence
			<p>IPR2021-00020 Petition for IPR at 1, 5, 19-20, 22-26, 28-49, 51-67, 72, 76-77;</p> <p>IPR2021-00020 Patent Owner's Preliminary Response at 1-2, 10-12, 14, 21, 25, 31, 33-46;</p> <p>IPR2021-00020 Denial of Institution of IPR at 5-7, 13-15, 20-24, 28.</p>		
<p>a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture (Claim 1)</p> <p>a mechanism for the nodes to communicate results of mathematical expression</p>	Claims 1, 29, 35	<p>Not subject to 35 U.S.C. § 112 ¶ 6; no construction necessary.</p> <p>Should the Court determine that these terms require construction:</p> <p>hardware and/or software modules that support internode communication using a peer-to-peer architecture</p> <p>Should the Court determine that § 112, ¶ 6 applies:</p>	<p>U.S. Patent No. 10,333,768 at col. 3, ll. 46-49, col. 5, l. 56-col. 6, l. 13, 22-25, col. 11, ll. 42-45, col. 13, l. 9-col. 14, l. 28, col. 14, ll. 36-43, col. 12, l. 33-col. 21, l. 10, col. 21, l. 63-col. 22, l. 24, col. 23, ll. 51-52, col. 24, l. 24-col. 25, l. 47, col. 26, ll. 9-32, FIG. 2, FIG. 3, FIG. 5, Table D, cls. 1-25, 30-35.</p> <p>File History for U.S. Patent No. 10,333,768 at 2014-09-08 Preliminary Amendment at 2; 2016-07-20 Non-Final</p>	<p>Subject to § 112 ¶ (6)</p> <p>Function: “communicate results of mathematical expression evaluation with each other (claim 29) [using a peer-to-peer architecture] (claim 1);</p> <p>“communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture” (claim 35)</p> <p>Structure (for claims 1, 29, 35): Message-Passing Interface</p>	<p>'768 patent at 6:3-8, 6:19-29, 11:42-54, 12:41-53, 13:10-15:9, 21:63-22:24, 24:58-67, 25:9-47, Figure 3.</p>

Claim Term	Claim(s)	ACS's Proposed Construction	ACS's Intrinsic Evidence	NVIDIA's Proposed Construction	NVIDIA's Intrinsic Evidence
<p>evaluation with each other (Claim 29)</p> <p>a mechanism to communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture (Claim 35)</p>		<p><u>Function:</u> communicate results of mathematical expression evaluation and user instructions between nodes</p> <p><u>Structure:</u> messaging modules that support communication using a peer-to-peer architecture</p>	<p>Rejection at 3-14; 2016-11-16 Applicant Initiated Interview Summary; 2016-11-16 Miscellaneous Internal Document; 2016-11-21 Claims at 2, 4-6; 2016-11-21 Applicant Arguments/Remarks Made in an Amendment at 8-9; 2017-03-09 Final Rejection at 3-6, 8, 13-15; 2017-08-08 Applicant Arguments/Remarks Made in an Amendment at 10-11; 2017-08-08 Miscellaneous Internal Document at 1; 2017-08-08 Applicant Initiated Interview Summary; 2018-04-13 Non-Final Rejection at 4-5, 7, 10, 15, 17-18; 2018-11-29 Issue Information including classification, examiner, name, claim renumbering etc. at 3. IPR2021-00019 Petition for IPR at 14, 28-32, 49-52, 57, 61, 64-70, 75-76,</p>	<p>("MPI") module 302, RMQ received message queue 306 and MRQ message receiving queue 308 configured to execute the process described at 25:29-43.</p>	

Claim Term	Claim(s)	ACS's Proposed Construction	ACS's Intrinsic Evidence	NVIDIA's Proposed Construction	NVIDIA's Intrinsic Evidence
			Declaration of Henry Tufo at 2-5, 24, 40; IPR2021-00019 Patent Owner's Preliminary Response at 1-3, 6-8, 20-26, 28, 30, 32, 42-47, 49-52; IPR2021-00019/IPR2021-00020 Petitioners Reply to Patent Owner's Preliminary Response; IPR2021-00019 Denial of Institution of IPR at 10-11, 14-15, 21-22, 21 n. 16; IPR2021-00020 Petition for IPR at 16-17, 28-34, 51-53, 60, 64-66, 73, Declaration of Henry Tufo at 1-5, 25, 77; IPR2021-00020 Patent Owner's Preliminary Response at 1-3, 6-12, 20-28, 30, 32, 42-46, 48-50; IPR2021-00020 Denial of Institution of IPR at 10-12, 15-16, 21-22, 21 n. 20.		
a mechanism for the nodes to communicate results of	Claim 26	Not subject to 35 U.S.C. § 112 ¶ 6; no construction necessary.	U.S. Patent No. 10,333,768 at col. 3, ll. 46-49, col. 5, l. 56-col. 6, l. 13, 19- 25, col. 11, ll.	Subject to § 112 ¶ (6) Function: "communicate results of	'768 patent at 6:3-8, 6:19-29, 11:42-54, 12:41-53, 13:10-15:9, 21:63-22:24, 24:58-67,

Claim Term	Claim(s)	ACS's Proposed Construction	ACS's Intrinsic Evidence	NVIDIA's Proposed Construction	NVIDIA's Intrinsic Evidence
mathematical expression evaluation with each other using asynchronous calls		<p>Should the Court determine that these terms require construction:</p> <p>hardware and/or software modules that support asynchronous internode communication using a peer-to-peer architecture</p> <p>Should the Court determine that § 112, ¶ 6 applies:</p> <p>Function: communicate results of mathematical expression evaluation and user instructions between nodes</p> <p>Structure: messaging modules that support asynchronous communication using a peer-to-peer architecture</p>	<p>42-45, col. 13, l. 9-col. 14, l. 43, col. 12, l. 33-col. 21, l. 10, col. 21, l. 63-col. 22, l. 24, col. 23, ll. 51-52, col. 24, l. 24-col. 25, l. 47, col. 26, ll. 9-32, col. 27, ll. 38-43, FIG. 2, FIG. 3, FIG. 5, Table C, Table D, cls. 1-28, 30-35.</p> <p>File History for U.S. Patent No. 10,333,768 at 2014-09-08 Preliminary Amendment at 2; 2016-07-20 Non-Final Rejection at 3-14; 2016-11-16 Applicant Initiated Interview Summary; 2016-11-16 Miscellaneous Internal Document; 2016-11-21 Claims at 2, 4-6; 2016-11-21 Applicant Arguments/Remarks Made in an Amendment at 8-9; 2017-03-09 Final Rejection at 3-6, 8, 10-17; 2017-08-08 Claims at 5; 2017-08-08 Applicant Arguments/Remarks Made in an Amendment</p>	<p>mathematical expression evaluation with each other using asynchronous calls”</p> <p>Structure: Message-Passing Interface (“MPI”) module 302, including mpiISend, mpiIRecv, and mpiTest commands as described at 14:1-28, RMQ received message queue 306 and MRQ message receiving queue 308 configured to execute the process described at 25:29-43.</p>	25:9-47, 27:39-48, Figure 3.

Claim Term	Claim(s)	ACS's Proposed Construction	ACS's Intrinsic Evidence	NVIDIA's Proposed Construction	NVIDIA's Intrinsic Evidence
			<p>at 10-11; 2017-08-08 Miscellaneous Internal Document at 1; 2017-08-08 Applicant Initiated Interview Summary; 2018-04-13 Non-Final Rejection at 4-5, 7, 10, 13-15, 17-18, 20-22, 25-26.</p> <p>IPR2021-00019 Petition for IPR at 1-2, 14, 28-32, 49-52, 57, 61, 64-78, Declaration of Henry Tufo at 2-5, 24, 40;</p> <p>IPR2021-00019 Patent Owner's Preliminary Response at 1-3, 6-8, 20-26, 28, 30, 32, 34-37, 42-47, 49-52;</p> <p>IPR2021-00019/IPR2021-00020 Petitioners Reply to Patent Owner's Preliminary Response; IPR2021-00019 Denial of Institution of IPR at 10-11, 14-15, 21-22, 21 n. 16;</p> <p>IPR2021-00020 Petition for IPR at 16-17, 28-34,</p>		

Claim Term	Claim(s)	ACS's Proposed Construction	ACS's Intrinsic Evidence	NVIDIA's Proposed Construction	NVIDIA's Intrinsic Evidence
			51-53, 60, 64-67, 73, Declaration of Henry Tufo at 1-5, 25, 77; IPR2021-00020 Patent Owner's Preliminary Response at 1-3, 6-12, 20-28, 30, 33-46, 48-50; IPR2021-00020 Denial of Institution of IPR at 10-12, 15-16, 21-22, 21 n. 20.		
wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second	Claims 1, 26, and 29	plain and ordinary meaning: wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and	U.S. Patent No. 10,333,768 at col. 3, ll. 39-55, col. 5, l. 56-col. 6, l. 25, col. 22, l. 48-col. 23, l. 7, FIG. 2, claims 1-39. File History for U.S. Patent No. 10,333,768 at 2017-08-08 Claims at 7-8; 2018-04-13 Non-Final Rejection at 26; 2018-10-05 Applicant Arguments/Remarks Made in an Amendment at 13-14; 2018-10-05 Claims at 10; 2018-11-29 Examiner Initiated Interview Summary; 2018-11-29 Notice of	This limitation requires a precise order of operations involving three specific nodes: (1) the third node receives, from the second node, a result of a calculation that was performed by the second node in a different claim limitation; (2) the third node performs a different calculation based on the received result; and (3) the third node sends the new result to the first node.	IPR2021-00019, Paper 5 at 26-27. IPR2021-00020, Paper 5 at 18-19.

Claim Term	Claim(s)	ACS's Proposed Construction	ACS's Intrinsic Evidence	NVIDIA's Proposed Construction	NVIDIA's Intrinsic Evidence
mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node		communicate the result of the second mathematical expression evaluation to the first node	<p>Allowance at 11-12; 2018-11-29 Index of Claims.</p> <p>IPR2021-00019 Petition for IPR at 38, 41-47;</p> <p>IPR2021-00019 Patent Owner's Preliminary Response at 26-29, 44-47, Singh Declaration at 21, 23, 34;</p> <p>IPR2021-00019 Denial of Institution of IPR at 17-18, 23-30;</p> <p>IPR2021-00020 Petition for IPR at 41-50, 63-64;</p> <p>IPR2021-00020 Patent Owner's Preliminary Response at 18-21, 43-45, Singh Declaration at 26;</p> <p>IPR2021-00020 Denial of Institution of IPR at 18, 24-29, 30-31.</p>		
node/nodes	Claims 1, 4, 7, 8, 10, 18-22, 24-27, 29-31, 33- 37, 39	<p>No construction necessary.</p> <p>Should the Court determine that these terms require</p>	U.S. Patent No. 10,333,768 at Abstract, col. 1, ll. 18-22, col. 1, ll. 37-41, col. 1, ll. 47-48, col. 1, ll. 52-62, col. 2, ll.	a processing unit or subunit that is capable of single-threaded execution of code	'768 patent: col 1: lines 19-28, 4:45-47, 4:63-5:35, 5:56-6:6, 7:1-32, 8:29-49, 9:48-67, 11:15-39,

Claim Term	Claim(s)	ACS's Proposed Construction	ACS's Intrinsic Evidence	NVIDIA's Proposed Construction	NVIDIA's Intrinsic Evidence
		<p>construction:</p> <p>Computing device[s] (e.g., computer[s], microprocessor[s], special purpose microprocessor[s], and/or processor core[s]) that can intercommunicate with other nodes</p>	<p>4-28, col. 2, ll. 18-21, col. 2, ll. 58-61, col. 3, ll. 13-26, col. 4, ll. 36-62, col. 5, ll. 33-55, col. 6, ll. 10-38, col. 7, ll. 10-32, col. 8, ll. 29-49, col. 9, ll. 59-67, col. 12, ll. 33-40, col. 12, ll. 55-60, col. 13, ll. 17-24, col. 6, ll. 41-47, col. 14, ll. 36-43, col. 15, ll. 10-17, col. 15, ll. 52-59, col. 17, ll. 12-8, col. 6, ll. 10-38, col. 18, ll. 35-48, col. 19, ll. 2-13, col. 20, ll. 2-14, col. 22, ll. 38-48, FIGS. 1-5, claims 1-39.</p> <p>IPR2021-00019 Patent Owner's Preliminary Response at 11-13, 20-26, Declaration of Henry Tufo at 24, 27, 51-52;</p> <p>IPR2021-00020 Patent Owner's Preliminary Response at 2. Declaration of Henry Tufo at 24-25.</p>		<p>Figs. 1-3.</p>

EXHIBIT B



US010333768B2

(12) **United States Patent**
Tannenbaum et al.

(10) **Patent No.:** **US 10,333,768 B2**

(45) **Date of Patent:** ***Jun. 25, 2019**

(54) **CLUSTER COMPUTING**

(71) Applicant: **Advanced Cluster Systems, Inc.**, Aliso Viejo, CA (US)

(72) Inventors: **Zvi Tannenbaum**, Newport Beach, CA (US); **Dean E. Dager**, Huntington Beach, CA (US)

(73) Assignee: **Advanced Cluster Systems, Inc.**, Aliso Viejo, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 524 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **14/181,112**

(22) Filed: **Feb. 14, 2014**

(65) **Prior Publication Data**

US 2014/0372586 A1 Dec. 18, 2014

Related U.S. Application Data

(63) Continuation of application No. 13/423,063, filed on Mar. 16, 2012, now Pat. No. 8,676,877, which is a (Continued)

(51) **Int. Cl.**
G06F 9/50 (2006.01)
G06F 9/54 (2006.01)

(Continued)

(52) **U.S. Cl.**
CPC **H04L 41/04** (2013.01); **G06F 9/5072** (2013.01); **G06F 9/54** (2013.01); **G06F 15/76** (2013.01)

(58) **Field of Classification Search**

USPC 709/223
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,423,046 A 6/1995 Nunnelley et al.
5,881,315 A 3/1999 Cohen
(Continued)

FOREIGN PATENT DOCUMENTS

JP 08-87473 A 2/1996
JP H11-126196 5/1999
(Continued)

OTHER PUBLICATIONS

"GridMathematica 1.1: Grid Computing Gets a Speed Boost from Mathematica 5", The Mathematica Journal, vol. 9, No. 2, 2004.

(Continued)

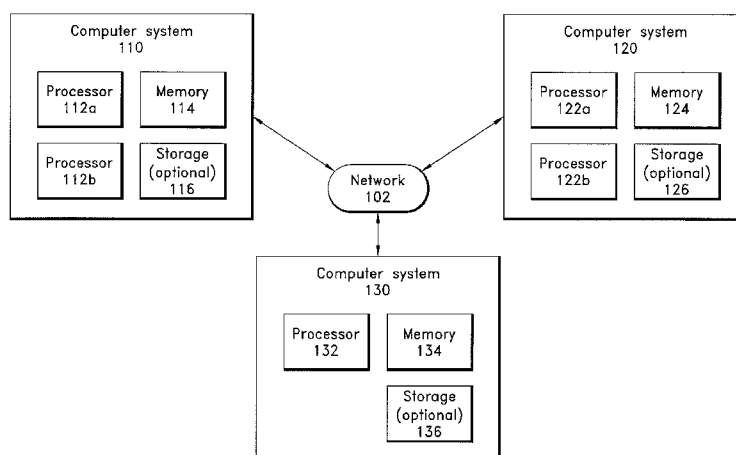
Primary Examiner — Hermon Asres

(74) *Attorney, Agent, or Firm* — Knobbe Martens Olson & Bear LLP

(57) **ABSTRACT**

In some embodiments, a computer cluster system comprises a plurality of nodes and a software package comprising a user interface and a kernel for interpreting program code instructions. In certain embodiments, a cluster node module is configured to communicate with the kernel and other cluster node modules. The cluster node module can accept instructions from the user interface and can interpret at least some of the instructions such that several cluster node modules in communication with one another and with a kernel can act as a computer cluster.

39 Claims, 5 Drawing Sheets



US 10,333,768 B2

Page 2

Related U.S. Application Data

continuation of application No. 12/040,519, filed on Feb. 29, 2008, now Pat. No. 8,140,612, which is a continuation-in-part of application No. 11/744,461, filed on May 4, 2007, now Pat. No. 8,082,289.

- (60) Provisional application No. 60/813,738, filed on Jun. 13, 2006, provisional application No. 60/850,908, filed on Oct. 11, 2006.

(51) Int. Cl.

G06F 15/76 (2006.01)

H04L 12/24 (2006.01)

(56)

References Cited

U.S. PATENT DOCUMENTS

5,930,768	A	7/1999	Hooban	
6,006,259	A	12/1999	Adelman et al.	
6,067,609	A	5/2000	Meeker et al.	
6,108,699	A	8/2000	Moin	
6,195,680	B1	2/2001	Goldszmidt et al.	
6,202,080	B1	3/2001	Lu et al.	
6,546,403	B1	4/2003	Carlson, Jr. et al.	
6,578,068	B1	6/2003	Bowman-Amuah	
6,604,134	B2	8/2003	Hauray	
6,751,698	B1	6/2004	Deneroff et al.	
6,782,537	B1	8/2004	Blackmore et al.	
6,968,335	B2	11/2005	Bayliss et al.	
6,968,359	B1 *	11/2005	Miller	G06F 9/5061 709/205
7,031,944	B2	4/2006	Tanioka	
7,093,004	B2	8/2006	Bernardin et al.	
7,136,924	B2	11/2006	Dauger	
7,139,882	B2	11/2006	Suzuoki et al.	
7,174,381	B2	2/2007	Gulko et al.	
7,177,874	B2	2/2007	Jardin	
7,249,357	B2	7/2007	Landman et al.	
7,315,877	B2	1/2008	Bhanot et al.	
7,334,232	B2	2/2008	Jacobs et al.	
7,437,460	B2	10/2008	Chidambaram et al.	
7,472,193	B2	12/2008	Dauger	
7,502,915	B2	3/2009	Jacob et al.	
7,519,652	B2	4/2009	Page et al.	
7,533,141	B2	5/2009	Nadgir et al.	
7,533,389	B2	5/2009	Verbeke et al.	
7,554,949	B2	6/2009	Chen	
7,568,131	B2	7/2009	Vertes	
7,644,130	B2	1/2010	Magro et al.	
7,698,390	B1	4/2010	Harkness et al.	
7,716,323	B2	5/2010	Gole et al.	
7,757,236	B1	7/2010	Singh	
7,788,314	B2	8/2010	Holt	
7,835,022	B2	11/2010	Matsumoto	
7,937,455	B2	5/2011	Saha et al.	
8,082,289	B2	12/2011	Tannenbaum et al.	
8,117,288	B2	2/2012	Bhanot et al.	
8,140,612	B2	3/2012	Tannenbaum et al.	
8,402,080	B2	3/2013	Tannenbaum et al.	
8,402,083	B2	3/2013	Tannenbaum et al.	
8,601,101	B1 *	12/2013	Singh	H04L 69/40 370/254
8,676,877	B2	3/2014	Tannenbaum et al.	
8,849,889	B1	9/2014	Tannenbaum et al.	
9,405,564	B2	8/2016	Muellers et al.	
2002/0049859	A1	4/2002	Bruckert et al.	
2003/0005068	A1	1/2003	Nickel et al.	
2003/0005266	A1	1/2003	Dauger	
2003/0051062	A1	3/2003	Circenis	
2003/0135621	A1	7/2003	Romagnoli	
2003/0195931	A1	10/2003	Dauger	
2003/0195938	A1 *	10/2003	Howard	G06F 8/45 709/208
2004/0015968	A1	1/2004	Neiman	
2004/0098359	A1	5/2004	Bayliss	

2004/0098447	A1	5/2004	Verbeke	
2004/0110209	A1	6/2004	Yokota et al.	
2004/0157203	A1	8/2004	Dunk	
2004/0195572	A1	10/2004	Kato et al.	
2004/0252710	A1	12/2004	Jeter, Jr. et al.	
2004/0254984	A1	12/2004	Dinker	
2005/0015460	A1	1/2005	Goyle et al.	
2005/0021751	A1 *	1/2005	Block	G06F 9/54 709/225
2005/0038852	A1	2/2005	Howard	
2005/0060237	A1	3/2005	Barsness et al.	
2005/0076105	A1	4/2005	Keohane et al.	
2005/0097300	A1	5/2005	Gildea et al.	
2005/0108394	A1	5/2005	Braun et al.	
2005/0154789	A1	7/2005	Fellenstein et al.	
2005/0180095	A1	8/2005	Ellis	
2005/0188088	A1	8/2005	Fellenstein et al.	
2006/0026601	A1	2/2006	Solt	
2006/0053216	A1	3/2006	Deokar et al.	
2006/0059473	A1	3/2006	Moler	
2006/0106931	A1	5/2006	Richoux	
2007/0044099	A1	2/2007	Rajput	
2007/0073705	A1 *	3/2007	Gray	G06F 9/451
2007/0094532	A1 *	4/2007	Sengupta	G06F 11/3632 714/5.1
2007/0124363	A1	5/2007	Lurie et al.	
2008/0250347	A1	10/2008	Gray et al.	
2008/0281997	A1	11/2008	Archer et al.	

FOREIGN PATENT DOCUMENTS

JP	11-328134	11/1999
JP	2002117010	4/2002
JP	2004-061359	2/2004
JP	2004-247405	9/2004
JP	2005-063033	10/2005
WF	1368948	12/2003
WO	WO 2004/086270	10/2004

OTHER PUBLICATIONS

“Wolfram gridMathematica”, Wolfram Research, Inc., 2007.

Carns et al., “An Evaluation of Message Passing Implementations on Beowulf Workstations”, Aerospace Conference, Mar. 6-13, 1999, IEEE 1999, vol. 5, pp. 41-54.

Dauger et al., “Plug-and-Play Cluster Computing using Mac OS X”, IEEE International Conference, Dec. 1-4, 2003, Paper appears in Cluster Computing Jan. 8, 2004, pp. 430-435.

Dauger et al., Plug-and-Play Cluster Computing: High-Performance Computing for the Mainstream, IEEE Computing in Science and Engineering, Mar./Apr. 2005, pp. 27-33.

Hamscher et al., “Evaluation of Job-Scheduling Strategies for grid Computing”, LNCS: Lecture Notes in Computer Science, 2000, pp. 191-202.

International Search Report dated Sep. 11, 2008, International Application No. PCT/US07/70585.

Jahanzeb et al., “Libra: a computational economy-based job scheduling system for clusters”, Software Practice and Experience, Feb. 24, 2004, vol. 34, pp. 573-590.

Jain et al., “Data Clustering: A Review”, ACM Computing Surveys, Sep. 1999, vol. 31, No. 3.

Kepner, Jeremy et al., “Parallel Matlab: The Next Generation”, Aug. 20, 2004, MIT Lincoln Laboratory, Lexington, MA.

Kim, Hahn et al., “Introduction to Parallel Programming and pMatlab v2.0”, 2011, MIT Lincoln Laboratory, Lexington, MA.

Maeder, R., “Mathematica Parallel Computing Toolkit: Unleash the Power of Parallel Computing”, Wolfram Research, Jan. 2005.

Tepeneu and Ida, “MathGridLink—A bridge between Mathematica and the Grid”, Nippon Sofutowa Kagakkai Taikai Ronbunshu, 2003, vol. 20, pp. 74-77.

Wolfram, Stephen: The Mathematica Book 5th Edition; Wolfram Research, Inc. 2003.

Haynos, Matt, “Perspectives on grid: Grid computing—next-generation distributed computing”, Jan. 27, 2004, IBM Developer Works, <http://www-106.ibm.com/developerworks/library/gr-heritage>.

US 10,333,768 B2

Page 3

(56)

References Cited

OTHER PUBLICATIONS

“Pingpong MPI Benchmark—SEM vs ‘grid’,” Jan. 2009, Dauger Research, Inc., <http://daugerresearch.com/pooch/mathematica/>.
Wolfram Research, “Mathematics Parallel Computing Toolkit”, Jan. 2005, pp. 1-95.

English translation of Matsumura, et al., “Construction of distributed computing system for large-scale matrices intended or reduction of communication blocks,” The Special Interest Group Technical Reports of IPSJ, Information Processing Society of Japan, Mar. 5, 1998, vol. 98, No. 18, pp. 19-24.

Konishi, et al., “Performance Evaluation of Parallel Computing Tool for MATLAB on PC Grid Environment,” IPSJ SIG Technical Reports, Aug. 5, 2005, vol. 2005, No. 84. 9 pages.

Matsumura, et al. “Construction of distributed computing system for large-scale matrices intended or reduction of communication blocks,” Aug. 5, 1998, vol. 98, No. 18. 9 pages.

Tatebe, et al., “Efficient Implementation of MPI Using Remote Memory Write,” Transactions of Information Processing Society of Japan, May 1999, vol. 40, No. 5. 14 pages.

U.S. Appl. No. 60/799,474, filed May 10, 2006, titled “Graphical Interface for Monitoring Status of a Concurrent Computing Process”.

Roman E. Maeder “Mathematica Parallel Computing Toolkit—Unleash the Power of Parallel Computing”, Jan. 1, 2005, pp. i-x, 1-90. URL:<http://media.wolfram.com/documents/ParallelComputingToolkitDocumentation.pdf>; retrieved on Aug. 21, 2015.
European Search Report received in Application No. 17207443.7, dated Sep. 20, 2018, in 9 pages.

* cited by examiner

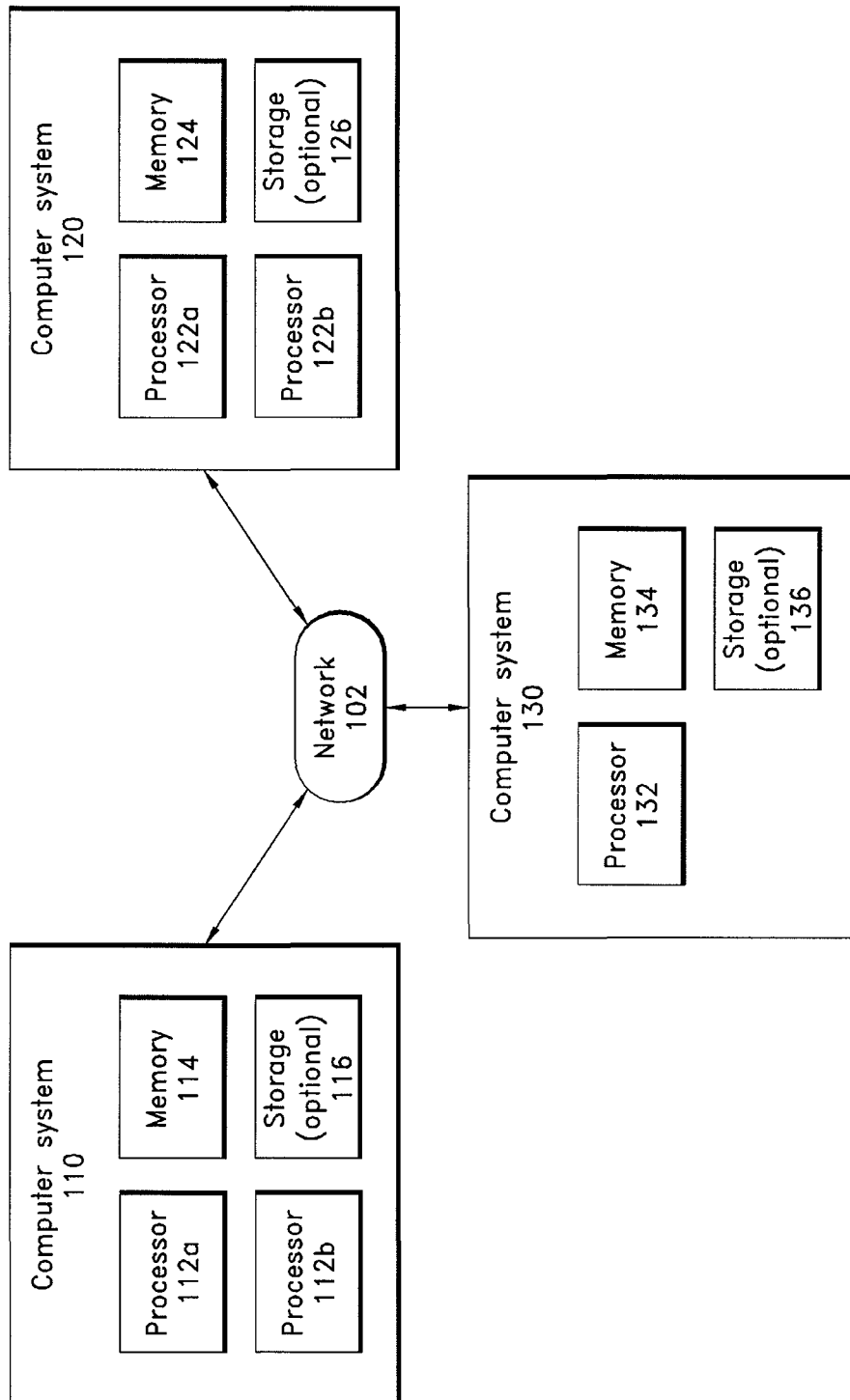


FIG. 1

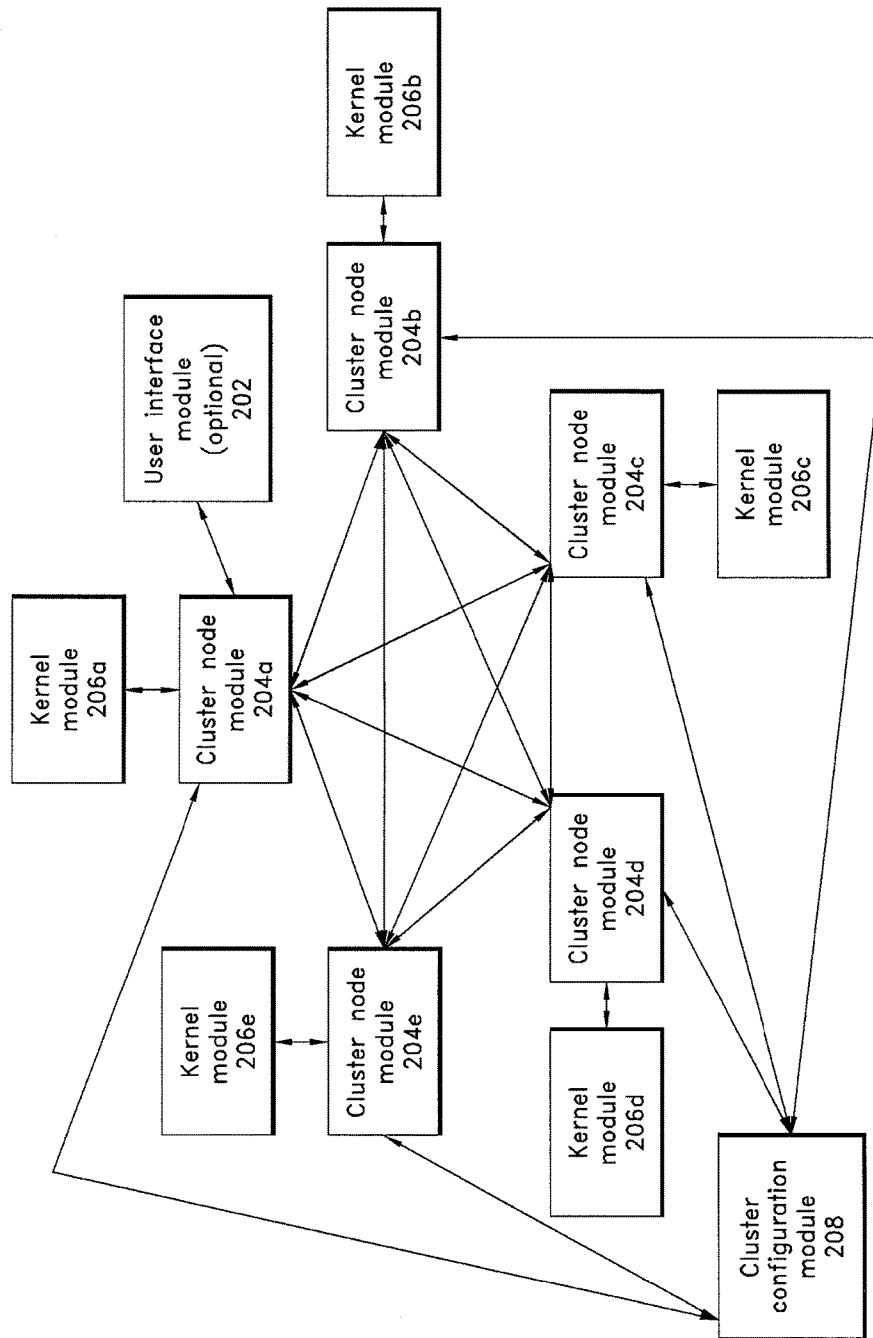


FIG. 2

U.S. Patent

Jun. 25, 2019

Sheet 3 of 5

US 10,333,768 B2

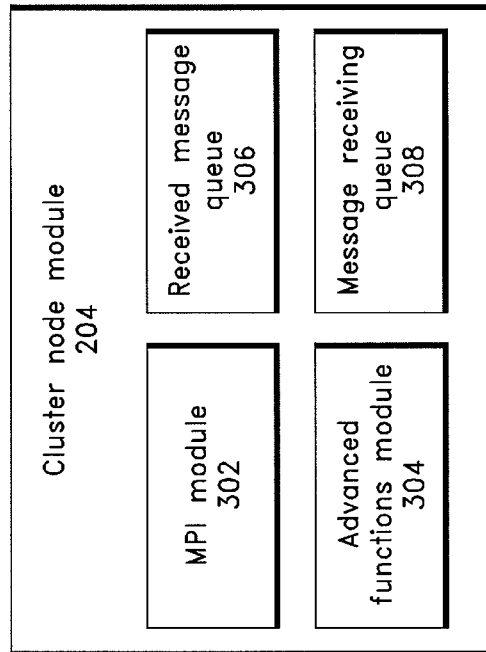
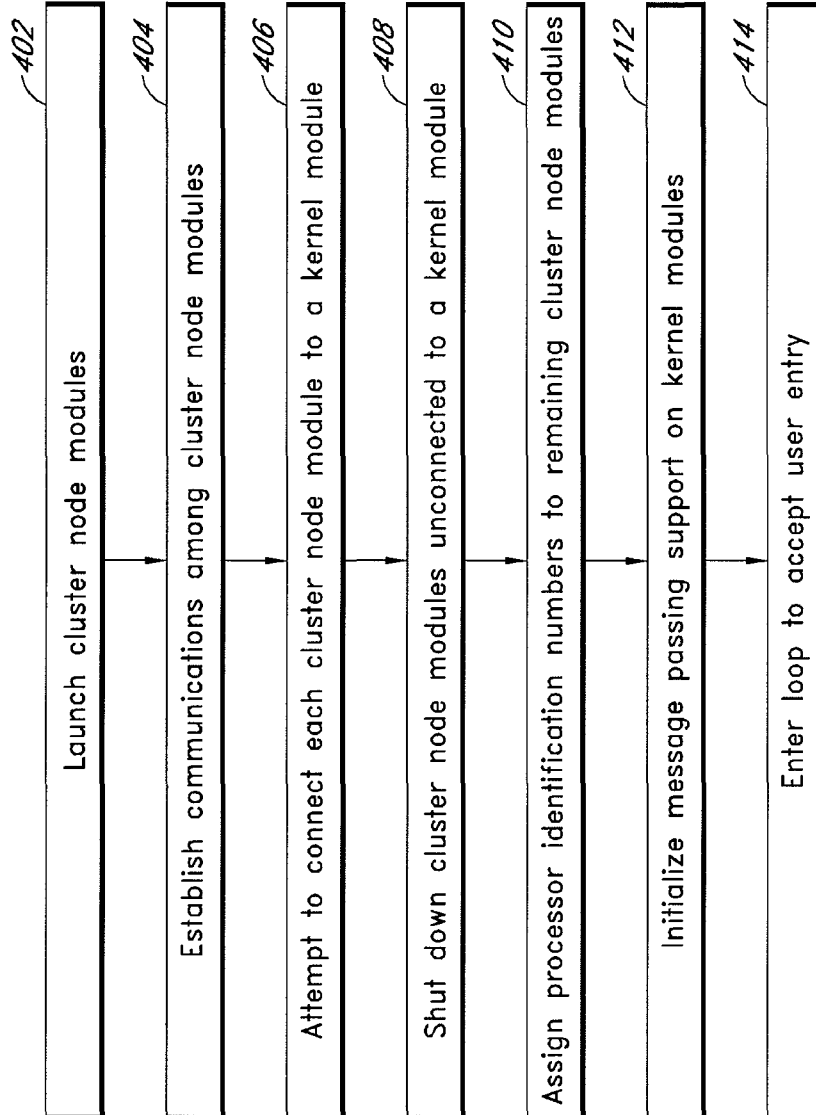
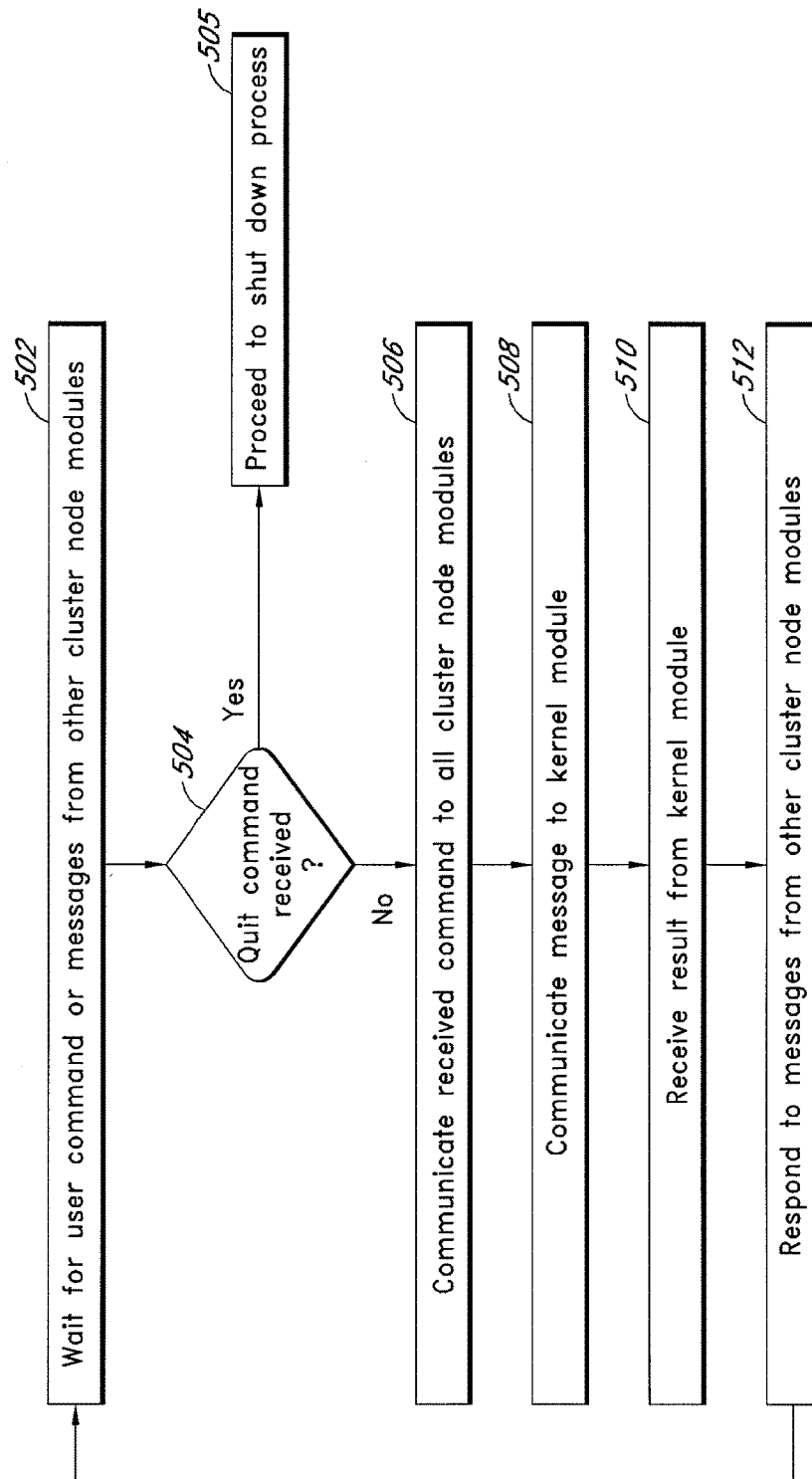


FIG. 3

*FIG. 4*

*FIG. 5*

US 10,333,768 B2

1

CLUSTER COMPUTING**INCORPORATION BY REFERENCE TO ANY
PRIORITY APPLICATIONS**

Any and all applications for which a foreign or domestic priority claim is identified in the Application Data Sheet as filed with the present application are incorporated by reference under 37 CFR 1.57 and made a part of this specification.

BACKGROUND**Field**

The present disclosure relates to the field of cluster computing generally and to systems and methods for adding cluster computing functionality to a computer program, in particular.

Description of Related Art

Computer clusters include a group of two or more computers, microprocessors, and/or processor cores ("nodes") that intercommunicate so that the nodes can accomplish a task as though they were a single computer. Many computer application programs are not currently designed to benefit from advantages that computer clusters can offer, even though they may be running on a group of nodes that could act as a cluster. Some computer programs can run on only a single node because, for example, they are coded to perform tasks serially or because they are designed to recognize or send instructions to only a single node.

Some application programs include an interpreter that executes instructions provided to the program by a user, a script, or another source. Such an interpreter is sometimes called a "kernel" because, for example, the interpreter can manage at least some hardware resources of a computer system and/or can manage communications between those resources and software (for example, the provided instructions, which can include a high-level programming language). Some software programs include a kernel that is designed to communicate with a single node. An example of a software package that includes a kernel that is designed to communicate with a single node is Mathematica® from Wolfram Research, Inc. ("Mathematica"). Mathematics software packages from other vendors and other types of software can also include such a kernel.

A product known as gridMathematica, also from Wolfram Research, Inc., gives Mathematica the capability to perform a form of grid computing known as "distributed computing." Grid computers include a plurality of nodes that generally do not communicate with one another as peers. Distributed computing can be optimized for workloads that consist of many independent jobs or packets of work, which do not need to share data between the jobs during the computational process. Grid computers include at least one node known as a master node that manages a plurality of slave nodes or computational nodes. In gridMathematica, each of a plurality of kernels runs on a single node. One kernel is designated the master kernel, which handles all input, output, and scheduling of the other kernels (the computational kernels or slave kernels). Computational kernels receive commands and data only from the node running the master kernel. Each computational kernel performs its work independently of the other computational kernels and intermediate results of one job do not affect other jobs in progress on other nodes.

SUMMARY

Embodiments described herein have several features, no single one of which is solely responsible for their desirable

2

attributes. Without limiting the scope of the invention as expressed by the claims, some of the advantageous features will now be discussed briefly.

Some embodiments described herein provide techniques for conveniently adding cluster computing functionality to a computer application. In one embodiment, a user of a software package may be able to achieve higher performance and/or higher availability from the software package by enabling the software to benefit from a plurality of nodes in a cluster. One embodiment allows a user to create applications, using a high-level language such as Mathematica, that are able to run on a computer cluster having supercomputer-like performance. One embodiment provides access to such high-performance computing through a Mathematica Front End, a command line interface, one or more high-level commands, or a programming language such as C or FORTRAN.

One embodiment adapts a software module designed to run on a single node, such as, for example, the Mathematica kernel, to support cluster computing, even when the software module is not designed to provide such support. One embodiment provides parallelization for an application program, even if no access to the program's source code is available. One embodiment adds and supports Message Passing Interface ("MPI") calls directly from within a user interface, such as, for example, the Mathematica programming environment. In one embodiment, MPI calls are added to or made available from an interactive programming environment, such as the Mathematica Front End.

One embodiment provides a computer cluster including a first processor, a second processor, and a third processor. The cluster includes at least one computer-readable medium in communication with at least one of the first processor, the second processor, or the third processor. A first kernel resides in the at least one computer-readable medium and is configured to translate commands into code for execution on the first processor. A first cluster node module resides in the at least one computer-readable medium. The first cluster node module is configured to send commands to the first kernel and receives commands from a user interface. A second kernel resides in the at least one computer-readable medium. The second kernel is configured to translate commands into code for execution on the second processor. A second cluster node module resides in the at least one computer-readable medium. The second cluster node module is configured to send commands to the second kernel and communicates with the first cluster node module. A third kernel resides in the at least one computer-readable medium. The third kernel is configured to translate commands into code for execution on the third processor. A third cluster node module resides in the at least one computer-readable medium. The third cluster node module is configured to send commands to the third kernel and configured to communicate with the first cluster node module and the second cluster node module. The first cluster node module comprises a data structure in which messages originating from the second and third cluster node modules are stored.

Another embodiment provides a computer cluster that includes a plurality of nodes and a software package including a user interface and a single-node kernel for interpreting program code instructions. A cluster node module is configured to communicate with the single-node kernel and other cluster node modules. The cluster node module accepts instructions from the user interface and interprets at least some of the instructions such that several cluster node modules in communication with one another act as a cluster. The cluster node module appears as a single-node kernel to

US 10,333,768 B2

3

the user interface. In one embodiment, the single-node kernel includes a Mathematica kernel. In some embodiments, the user interface can include at least one of a Mathematica front end or a command line. In some embodiments, the cluster node module includes a toolkit including library calls that implement at least a portion of MPI calls. In some embodiments, the cluster node module includes a toolkit including high-level cluster computing commands. In one embodiment, the cluster system can include a plurality of Macintosh® computers ("Macs"), Windows®-based personal computers ("PCs"), and/or Unix/Linux-based workstations.

A further embodiment provides a computer cluster including a plurality of nodes. Each node is configured to access a computer-readable medium comprising program code for a user interface and program code for a single-node kernel module configured to interpret user instructions. The cluster includes a plurality of cluster node modules. Each cluster node module is configured to communicate with a single-node kernel and with one or more other cluster node modules, to accept instructions from the user interface, and to interpret at least some of the user instructions such that the plurality of cluster node modules communicate with one another in order to act as a cluster. A communications network connects the nodes. One of the plurality of cluster node modules returns a result to the user interface.

Another embodiment provides a method of evaluating a command on a computer cluster. A command from at least one of a user interface or a script is communicated to one or more cluster node modules within the computer cluster. Each of the one or more cluster node modules communicates a message based on the command to a respective kernel module associated with the cluster node module. Each of the one or more cluster node modules receives a result from the respective kernel module associated with the cluster node module. At least one of the one or more cluster node modules responds to messages from other cluster node modules.

Another embodiment provides a computing system for executing Mathematica code on multiple nodes. The computing system includes a first node module in communication with a first Mathematica kernel executing on a first node, a second node module in communication with a second Mathematica kernel executing on a second node, and a third node module in communication with a third Mathematica kernel executing on a third node. The first node module, the second node module, and the third node module are configured to communicate with one another using a peer-to-peer architecture. In some embodiments, each of the first node module, the second node module, and third node module includes a data structure for maintaining messages originating from other node modules and a data structure for maintaining data specifying a location to which a message is expected to be received and an identifier for a node from which the message is expected to be sent.

BRIEF DESCRIPTION OF THE DRAWINGS

A general architecture that implements the various features are described with reference to the drawings. The drawings and the associated descriptions are provided to illustrate embodiments and not to limit the scope of the disclosure. Throughout the drawings, reference numbers are re-used to indicate correspondence between referenced elements.

FIG. 1 is a block diagram of one embodiment of a computer cluster.

4

FIG. 2 is a block diagram showing relationships between software modules running on one embodiment of a computer cluster.

FIG. 3 is a block diagram of one embodiment of a cluster node module.

FIG. 4 is a flow chart showing one embodiment of a cluster initialization process.

FIG. 5 is a flow chart showing one embodiment of the operation of a cluster node module.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

For purposes of illustration, some embodiments are described herein in the context of cluster computing with Mathematica software. The present disclosure is not limited to a single software program; the systems and methods can be used with other application software such as, for example, Maple®, MATLAB®, MathCAD®, Apple Shake®, Apple® Compressor, IDL®, other applications employing an interpreter or a kernel, Microsoft Excel®, Adobe After Effects®, Adobe Premiere®, Adobe Photoshop®, Apple Final Cut Pro®, and Apple iMovie®. Some figures and/or descriptions, however, relate to embodiments of computer clusters running Mathematica. The system can include a variety of uses, including but not limited to students, educators, scientists, engineers, mathematicians, researchers, and technicians. It is also recognized that in other embodiments, the systems and methods can be implemented as a single module and/or implemented in conjunction with a variety of other modules. Moreover, the specific implementations described herein are set forth in order to illustrate, and not to limit, the disclosure.

I. Overview

The cluster computing system described herein generally includes one or more computer systems connected to one another via a communications network or networks. The communications network can include one or more of a local area network ("LAN"), a wide area network ("WAN"), an intranet, the Internet, etc. In one embodiment, a computer system comprises one or more processors such as, for example, a microprocessor that can include one or more processing cores ("nodes"). The term "node" refers to a processing unit or subunit that is capable of single-threaded execution of code. The processors can be connected to one or more memory devices such as, for example, random access memory ("RAM"), and/or one or more optional storage devices such as, for example, a hard disk. Communications among the processors and such other devices may occur, for example, via one or more local buses of a computer system or via a LAN, a WAN, a storage area network ("SAN"), and/or any other communications network capable of carrying signals among computer system components. In one embodiment, one or more software modules such as kernels, run on nodes within the interconnected computer systems. In one embodiment, the kernels are designed to run on only a single node. In one embodiment, cluster node modules communicate with the kernels and with each other in order to implement cluster computing functionality.

FIG. 1 is a block diagram of one embodiment of a computer cluster 100 wherein computer systems 110, 120, 130 communicate with one another via a communications network 102. Network 102 includes one or more of a LAN, a WAN, a wireless network, an intranet, or the Internet. In

US 10,333,768 B2

5

one embodiment of the computer cluster, computer system **110** includes processors **112a**, **112b**, memory **114**, and optional storage **116**. Other computer systems **120**, **130** can include similar devices, which generally communicate with one another within a computer system over a local communications architecture such as a local bus (not shown). A computer system can include one or more processors, and each processor can contain one or more processor cores that are capable of single-threaded execution. Processor cores are generally independent microprocessors, but more than one can be included in a single chip package. Software code designed for single-threaded execution can generally run on one processor core at a time. For example, single-threaded software code typically does not benefit from multiple processor cores in a computer system.

FIG. 2 is a block diagram showing relationships among software modules running on one embodiment of a computer cluster **100**. In the embodiment shown in FIG. 2, the kernel modules **206a-e** are designed for single-threaded execution. For example, if each of the processors **112a**, **112b**, **122a**, **122b**, **132** shown in FIG. 1 includes only one processor core, two kernel modules (for example, kernel modules **206a**, **206b**) loaded into the memory **114** of computer system **110** could exploit at least some of the processing bandwidth of the two processors **112a**, **112b**. Similarly, two kernel modules **206c**, **206d** loaded into the memory **124** of computer system **120** could exploit at least some of the processing bandwidth of the two processors **122a**, **122b**. Likewise, the bandwidth of processor **132** of computer system **130** could be utilized by a single instance of a cluster node module **204e** loaded into the computer system's memory **134**.

In the embodiment shown in FIG. 2, each of the kernel modules **206a-e** is in communication with a single cluster node module **204a-e**, respectively. For example, the kernel module **206a** is in communication with the cluster node module **204a**, the kernel module **206b** is in communication with the cluster node module **206b**, and so forth. In one embodiment, one instance of a cluster node module **204a-e** is loaded into a computer system's memory **114**, **124**, **134** for every instance of a kernel module **206a-e** running on the system. As shown in FIG. 2, each of the cluster node modules **204a-e** is in communication with each of the other cluster node modules **204a-e**. For example, one cluster node module **204a** is in communication with all of the other cluster node modules **204b-e**. A cluster node module **204a** may communicate with another cluster node module **204b** via a local bus (not shown) when, for example, both cluster node modules **204a-b** execute on processors **112a**, **112b** within the same computer system **110**. A cluster node module **204a** may also communicate with another cluster node module **204c** over a communications network **102** when, for example, the cluster node modules **204a**, **c** execute on processors **112a**, **122a** within different computer systems **110**, **120**.

As shown in FIG. 2, an optional user interface module **202** such as, for example, a Mathematica front end and/or a command line interface, can connect to a cluster node module **204a**. The user interface module can run on the same computer system **110** and/or the same microprocessor **112a** on which the cluster node module **204a** runs. The cluster node modules **204a-e** provide MPI calls and/or advanced cluster functions that implement cluster computing capability for the single-threaded kernel modules. The cluster node modules **204a-e** are configured to look and behave like a kernel module **206a** from the perspective of the user interface module **202**. Similarly, the cluster node modules **204a-e**

6

are configured to look and behave like a user interface module **202** from the perspective of a kernel module **206a**. The first cluster node module **204a** is in communication with one or more other cluster node modules **204b**, **204c**, and so forth, each of which provides a set of MPI calls and/or advanced cluster commands. In one embodiment, MPI may be used to send messages between nodes in a computer cluster.

Communications can occur between any two or more cluster node modules (for example, between a cluster node module **204a** and another cluster node module **204c**) and not just between "adjacent" kernels. Each of the cluster node modules **204a-e** is in communication with respective kernel modules **206a-e**. Thus, the cluster node module **204a** communicates with the kernel module **206a**. MPI calls and advanced cluster commands are used to parallelize program code received from an optional user interface module **208** and distribute tasks among the kernel modules **206a-e**. The cluster node modules **204a-e** provide communications among kernel modules **206a-e** while the tasks are executing. Results of evaluations performed by kernel modules **206a-e** are communicated back to the first cluster node module **204a** via the cluster node modules **204a-e**, which communicates them to the user interface module **208**.

Intercommunication among kernel modules **206a-e** during thread execution, which is made possible by cluster node modules **204a-e**, provides advantages for addressing various types of mathematic and scientific problems, for example. Intercommunication provided by cluster computing permits exchange of information between nodes during the course of a parallel computation. Embodiments of the present disclosure provide such intercommunication for software programs such as Mathematica, while grid computing solutions can implement communication between only one master node and many slave nodes. Grid computing does not provide for communication between slave nodes during thread execution.

For purposes of providing an overview of some embodiments, certain aspects, advantages, benefits, and novel features of the invention are described herein. It is to be understood that not necessarily all such advantages or benefits can be achieved in accordance with any particular embodiment of the invention. Thus, for example, those skilled in the art will recognize that the invention can be embodied or carried out in a manner that achieves one advantage or group of advantages as taught herein without necessarily achieving other advantages or benefits as can be taught or suggested herein.

II. Computer Cluster 100

As shown in FIG. 1, one embodiment of a cluster system **100** includes computer systems **110**, **120**, **130** in communication with one another via a communications network **102**. A first computer system **110** can include one or more processors **112a-b**, a memory device **114**, and an optional storage device **116**. Similarly, a second computer system **120** can include one or more processors **122a-b**, a memory device **124**, and an optional storage device **126**. Likewise, a third computer system **130** can include one or more processors **132**, a memory device **134**, and an optional storage device **136**. Each of the computer systems **110**, **120**, **130** includes a network interface (not shown) for connecting to a communications network **102**, which can include one or more of a LAN, a WAN, an intranet, a wireless network, and/or the Internet.

US 10,333,768 B2

7

A. Computer System 110

In one embodiment, a first computer system **110** communicates with other computer systems **120**, **130** via a network **102** as part of a computer cluster **100**. In one embodiment, the computer system **110** is a personal computer, a workstation, a server, or a blade including one or more processors **112a-b**, a memory device **114**, an optional storage device **116**, as well as a network interface module (not shown) for communications with the network **102**.

1. Processors 112a-b

In one embodiment, the computer system **110** includes one or more processors **112a-b**. The processors **112a-b** can be one or more general purpose single-core or multi-core microprocessors such as, for example, a Pentium® processor, a Pentium® II processor, a Pentium® Pro processor, a Pentium® III processor, Pentium® 4 processor, a Core Duo® processor, a Core 2 Duo® processor, a Xeon® processor, an Itanium® processor, a Pentium® M processor, an x86 processor, an Athlon® processor, an 8051 processor, a MIPS® processor, a PowerPC® processor, an ALPHA® processor, etc. In addition, one or more of the processors **112a-b** can be a special purpose microprocessor such as a digital signal processor. The total number of processing cores (for example, processing units capable of single-threaded execution) within all processors **112a-b** in the computer system **110** corresponds to the number of nodes available in the computer system **110**. For example, if the processors **112a-b** were each Core 2 Duo® processors having two processing cores, computer system **110** would have four nodes in all. Each node can run one or more instances of a program module, such as a single-threaded kernel module.

2. Network Interface Module

The computer system **110** can also include a network interface module (not shown) that facilitates communication between the computer system **110** and other computer systems **120**, **130** via the communications network **102**.

The network interface module can use a variety of network protocols. In one embodiment, the network interface module includes TCP/IP. However, it is to be appreciated that other types of network communication protocols such as, for example, Point-to-Point Protocol ("PPP"), Server Message Block ("SMB"), Serial Line Internet Protocol ("SLIP"), tunneling PPP, AppleTalk, etc., may also be used.

3. Memory 114 and Storage 116

The computer system **110** can include memory **114**. Memory **114** can include, for example, processor cache memory (such as processor core-specific or cache memory shared by multiple processor cores), dynamic random-access memory ("DRAM"), static random-access memory ("SRAM"), or any other type of memory device capable of storing computer data, instructions, or program code. The computer system **110** can also include optional storage **116**. Storage **116** can include, for example, one or more hard disk drives, floppy disks, flash memory, magnetic storage media, CD-ROMs, DVDs, optical storage media, or any other type of storage device capable of storing computer data, instructions, and program code.

4. Computer System 110 Information

The computer system **110** may be used in connection with various operating systems such as: Microsoft® Windows® 3.X, Windows 95®, Windows 98®, Windows NT®, Windows 2000®, Windows XP®, Windows CE®, Palm Pilot OS, OS/2, Apple® MacOS®, MacOS X®, MacOS X Server®, Disk Operating System (DOS), UNIX, Linux®, VxWorks, or IBM® OS/2®, Sun OS, Solaris OS, IRIX OS operating systems, etc.

8

In one embodiment, the computer system **110** is a personal computer, a laptop computer, a BlackBerry® device, a portable computing device, a server, a computer workstation, a local area network of individual computers, an interactive kiosk, a personal digital assistant, an interactive wireless communications device, a handheld computer, an embedded computing device, or the like.

As can be appreciated by one of ordinary skill in the art, the computer system **110** may include various sub-routines, procedures, definitional statements, and macros. Each of the foregoing modules are typically separately compiled and linked into a single executable program. However, it is to be appreciated by one of ordinary skill in the art that the processes that are performed by selected ones of the modules may be arbitrarily redistributed to one of the other modules, combined together in a single module, made available in a shareable dynamic link library, or partitioned in any other logical way.

B. Computer System 120

In one embodiment, a second computer system **120** communicates with other computer systems **110**, **130** via a network **102** as part of a computer cluster **100**. In one embodiment, the computer system **120** is a personal computer, a workstation, a server, or a blade including one or more processors **122a-b**, a memory device **124**, an optional storage device **126**, as well as a network interface module (not shown) for communications with the network **102**.

1. Processors 122a-b

In one embodiment, the computer system **120** includes one or more processors **122a-b**. The processors **122a-b** can be one or more general purpose single-core or multi-core microprocessors such as a Pentium® processor, a Pentium® II processor, a Pentium® Pro processor, a Pentium® III processor, Pentium® 4 processor, a Core Duo® processor, a Core 2 Duo® processor, a Xeon® processor, an Itanium® processor, a Pentium® M processor, an x86 processor, an Athlon® processor, an 8051 processor, a MIPS® processor, a PowerPC® processor, an ALPHA® processor, etc. In addition, the processors **122a-b** can be any special purpose microprocessors such as a digital signal processor. The total number of processing cores (for example, processing units capable of single-threaded execution) within all processors **122a-b** in the computer system **120** corresponds to the number of nodes available in the computer system **120**. For example, if the processors **122a-b** were each Core 2 Duo® processors having two processing cores, computer system **120** would have four nodes in all. Each node can run one or more instances of a program module, such as a single-threaded kernel module.

2. Network Interface Module

The computer system **120** can also include a network interface module (not shown) that facilitates communication between the computer system **120** and other computer systems **110**, **130** via the communications network **102**.

The network interface module can use a variety of network protocols. In one embodiment, the network interface module includes TCP/IP. However, it is to be appreciated that other types of network communication protocols such as, for example, Point-to-Point Protocol ("PPP"), Server Message Block ("SMB"), Serial Line Internet Protocol ("SLIP"), tunneling PPP, AppleTalk, etc., may also be used.

3. Memory 124 and Storage 126

The computer system **120** can include memory **124**. Memory **124** can include, for example, processor cache memory (such as processor core-specific or cache memory shared by multiple processor cores), dynamic random-access memory ("DRAM"), static random-access memory

US 10,333,768 B2

9

("SRAM"), or any other type of memory device capable of storing computer data, instructions, or program code. The computer system **120** can also include optional storage **126**. Storage **126** can include, for example, one or more hard disk drives, floppy disks, flash memory, magnetic storage media, CD-ROMs, DVDs, optical storage media, or any other type of storage device capable of storing computer data, instructions, and program code.

4. Computer System **120** Information

The computer system **120** may be used in connection with various operating systems such as: Microsoft® Windows® 3.X, Windows 95®, Windows 98®, Windows NT®, Windows 2000®, Windows XP®, Windows CE®, Palm Pilot OS, OS/2, Apple® MacOS®, MacOS X®, MacOS X Server®, Disk Operating System (DOS), UNIX, Linux®, VxWorks, or IBM® OS/2®, Sun OS, Solaris OS, IRIX OS operating systems, etc.

In one embodiment, the computer system **120** is a personal computer, a laptop computer, a Blackberry® device, a portable computing device, a server, a computer workstation, a local area network of individual computers, an interactive kiosk, a personal digital assistant, an interactive wireless communications device, a handheld computer, an embedded computing device, or the like.

As can be appreciated by one of ordinary skill in the art, the computer system **120** may include various sub-routines, procedures, definitional statements, and macros. Each of the foregoing modules are typically separately compiled and linked into a single executable program. However, it is to be appreciated by one of ordinary skill in the art that the processes that are performed by selected ones of the modules may be arbitrarily redistributed to one of the other modules, combined together in a single module, made available in a shareable dynamic link library, or partitioned in any other logical way.

C. Computer System **130**

In one embodiment, a third computer system **130** communicates with other computer systems **110**, **120** via a network **102** as part of a computer cluster **100**. In one embodiment, the computer system **130** is a personal computer, a workstation, a server, or a blade including one or more processors **132**, a memory device **134**, an optional storage device **136**, as well as a network interface module (not shown) for communications with the network **102**.

1. Processors **112a-b**

In one embodiment, the computer system **130** includes a processor **132**. The processor **132** can be a general purpose single-core or multi-core microprocessors such as a Pentium® processor, a Pentium® II processor, a Pentium® Pro processor, a Pentium® III processor, Pentium® 4 processor, a Core Duo® processor, a Core 2 Duo® processor, a Xeon® processor, an Itanium® processor, a Pentium® M processor, an x86 processor, an Athlon® processor, an 8051 processor, a MIPS® processor, a PowerPC® processor, or an ALPHA® processor. In addition, the processor **132** can be any special purpose microprocessor such as a digital signal processor. The total number of processing cores (for example, processing units capable of single-threaded execution) within processor **132** in the computer system **130** corresponds to the number of nodes available in the computer system **130**. For example, if the processor **132** was a Core 2 Duo® processor having two processing cores, the computer system **130** would have two nodes. Each node can run one or more instances of a program module, such as a single-threaded kernel module.

10

2. Network Interface Module

The computer system **130** can also include a network interface module (not shown) that facilitates communication between the computer system **130** and other computer systems **110**, **120** via the communications network **102**.

The network interface module can use a variety of network protocols. In one embodiment, the network interface module includes TCP/IP. However, it is to be appreciated that other types of network communication protocols such as, for example, Point-to-Point Protocol ("PPP"), Server Message Block ("SMB"), Serial Line Internet Protocol ("SLIP"), tunneling PPP, AppleTalk, etc., may also be used.

3. Memory **134** and Storage **136**

The computer system **130** can include memory **134**. Memory **134** can include, for example, processor cache memory (such as processor core-specific or cache memory shared by multiple processor cores), dynamic random-access memory ("DRAM"), static random-access memory ("SRAM"), or any other type of memory device capable of storing computer data, instructions, or program code. The computer system **130** can also include optional storage **136**. Storage **136** can include, for example, one or more hard disk drives, floppy disks, flash memory, magnetic storage media, CD-ROMs, DVDs, optical storage media, or any other type of storage device capable of storing computer data, instructions, and program code.

4. Computer System **130** Information

The computer system **130** may be used in connection with various operating systems such as: Microsoft® Windows® 3.X, Windows 95®, Windows 98®, Windows NT®, Windows 2000®, Windows XP®, Windows CE®, Palm Pilot OS, OS/2, Apple® MacOS®, MacOS X®, MacOS X Server®, Disk Operating System (DOS), UNIX, Linux®, VxWorks, or IBM® OS/2®, Sun OS, Solaris OS, IRIX OS operating systems, etc.

In one embodiment, the computer system **130** is a personal computer, a laptop computer, a Blackberry® device, a portable computing device, a server, a computer workstation, a local area network of individual computers, an interactive kiosk, a personal digital assistant, an interactive wireless communications device, a handheld computer, an embedded computing device, or the like.

As can be appreciated by one of ordinary skill in the art, the computer system **130** may include various sub-routines, procedures, definitional statements, and macros. Each of the foregoing modules are typically separately compiled and linked into a single executable program. However, it is to be appreciated by one of ordinary skill in the art that the processes that are performed by selected ones of the modules may be arbitrarily redistributed to one of the other modules, combined together in a single module, made available in a shareable dynamic link library, or partitioned in any other logical way.

E. Communications Network **102**

In one embodiment, computer systems **110**, **120**, **130** are in communication with one another via a communications network **102**.

The communications network **102** may include one or more of any type of electronically connected group of computers including, for instance, the following networks: a virtual private network, a public Internet, a private Internet, a secure Internet, a private network, a public network, a value-added network, a wired network, a wireless network, an intranet, etc. In addition, the connectivity to the network can be, for example, a modem, Ethernet (IEEE 802.3), Gigabit Ethernet, 10-Gigabit Ethernet, Token Ring (IEEE 802.5), Fiber Distributed Datalink Interface (FDDI), Frame

11

Relay, InfiniBand, Myrinet, Asynchronous Transfer Mode (ATM), or another interface. The communications network **102** may connect to the computer systems **110**, **120**, **130**, for example, by use of a modem or by use of a network interface card that resides in each of the systems.

In addition, the same or different communications networks **102** may be used to facilitate communication between the first computer system **110** and the second computer system **120**, between the first computer system **110** and the third computer system **130**, and between the second computer system **120** and the third computer system **130**.

III. Software Modules

As shown in FIGS. **1** and **2**, one embodiment of a cluster system **100** includes a user interface module **202** that is able to access a plurality of kernel modules **206a-e** by communicating with a first cluster node module **204a**. User interface module can be stored in a memory **114**, **124**, **134** while running, for example, and/or can be stored in a storage device **116**, **126**, **136**. The first cluster node module **204a** is in communication with each of the other cluster node modules **204b-e**. The kernel modules **206a-e** can reside in the memory of one or more computer systems on which they run. For example, the memory **114** of the first computer system **110** can store instances of kernel modules **206a-b**, the memory **124** of the second computer system **120** can store instances of kernel modules **206c-d**, and the memory **134** of the third computer system **130** can store an instance of kernel module **206e**. The kernel modules **206a-e**, which include single-threaded program code, are each associated with one of the processors **112a**, **112b**, **122a**, **122b**, **132**. A cluster configuration module stored on one or more of the computer systems **110**, **120**, **130** or on a remote computer system, for example, can establish communication with the cluster node modules **204a-e**. In one embodiment, communication between the cluster configuration module **208** and the cluster node modules **204a-e** initializes the cluster node modules **204a-e** to provide cluster computing support for the computer cluster **100**.

A. Cluster Node Module **204**

In one embodiment, the cluster node modules **204a-e** provide a way for many kernel modules **206a-e** such as, for example, Mathematica kernels, running on a computer cluster **100** to communicate with one another. A cluster node module **204** can include at least a portion of an application programming interface (“API”) known as the Message-Passing Interface (“MPI”), which is used in several super-computer and cluster installations. A network of connections (for example, the arrows shown in FIG. **2**) between the cluster node modules **204a-e** can be implemented using a communications network **102**, such as, for example, TCP/IP over Ethernet, but the connections could also occur over any other type of network or local computer bus.

A cluster node module **204** can use an application-specific toolkit or interface such as, for example, Mathematica’s MathLink, Add-Ons, or packets, to interact with an application. Normally used to connect a Mathematica kernel to a user interface known as the Mathematica Front End or other Mathematica kernels, MathLink is a bidirectional protocol to sends “packets” containing messages, commands, or data between any of these entities. MathLink does not allow direct cluster computing-like simultaneous communication between Mathematica kernels during execution of a command or thread. MathLink is also not designed to perform multiple simultaneous network connections. In some embodiments, a cluster node module **204** can use an appli-

12

cation-specific toolkit such as, for example, MathLink, for connections between entities on the same computer.

When speaking about procedures or actions on a cluster or other parallel computer, not all actions happen in sequential order, nor are they required to. For example, a parallel code, as opposed to a single-processor code of the classic “Turing machine” model, has multiple copies of the parallel code running across the cluster, typically one for each processor (or “processing element” or “core”). Such parallel code is written in such a way that different instances of the same code can communicate, collaborate, and coordinate work with each other. Multiple instances of these codes can run at the same time in parallel.

If the count of the code instances is an integer N , each instance of code execution can be labeled 0 through $N-1$. For example, a computer cluster can include N connected computers, each containing a processor. The first has cluster node module 0 connected with kernel module 0 running on processor 0 . The next is cluster node module 1 and kernel module 1 , on processor 1 , and so forth for each of the N connected computers. Some steps of their procedure are collaborative, and some steps are independent. Even though these entities are not necessarily in lock-step, they do follow a pattern of initialization, main loop behavior (for example, cluster node module operation), and shut down.

In contrast, a parallel computing toolkit (PCT) that is provided as part of the gridMathematica software package does not provide a means for instances of the same code running on different nodes to communicate, collaborate, or coordinate work among the instances. The PCT provides commands that connect Mathematica kernels in a master-slave relationship rather than a peer-to-peer relationship as enabled by some embodiments disclosed herein. A computer cluster having peer-to-peer node architecture performs computations that can be more efficient, easier to design, and/or more reliable than similar computations performed on grid computers having master-slave node architecture. Moreover, the nature of some computations may not allow a programmer to harness multi-node processing power on systems that employ master-slave node architecture.

FIG. **3** shows one embodiment of a cluster node module **204** implementing MPI calls and advanced MPI functions. In the embodiment shown in FIG. **3**, cluster node module **204** includes MPI module **302**, advanced functions module **304**, received message queue **306**, and message receiving queue **308**.

1. MPI module **302**

In one embodiment, the cluster node module **204** includes an MPI module **302**. The MPI module **302** can include program code for one or more of at least five kinds of MPI instructions or calls. Selected constants, instructions, and/or calls that can be implemented by the MPI module **302** are as follows:

MPI Constants

Node identifiers are used to send messages to nodes or receive messages from them. In MPI, this is accomplished by assigning each node a unique integer (\$IdProc) starting with 0 . This data, with a knowledge of the total count (\$NProc), makes it possible to programmatically divide any measurable entity.

TABLE A

Constant	Description
\$IdProc	The identification number of the current processor
\$NProc	The number of processors in the current cluster

US 10,333,768 B2

13

TABLE A-continued

Constant	Description
\$mpiCommWorld	The communicator world of the entire cluster (see MPI Communicator routines, below)
mpiCommWorld	The default communicator world for the high-level routines.

Basic MPI Calls

In one embodiment, the MPI module 302 can include basic MPI calls such as, for example, relatively low-level routines that map MPI calls that are commonly used in other languages (such as C and Fortran), so that such calls can be available directly from the Mathematica user interface 204. In some embodiments, basic MPI calls include calls that send data, equations, formulas, and/or other expressions.

Simply sending expressions from one node to another is possible with these most basic MPI calls. One node can call to send an expression while the other calls a corresponding routine to receive the sent expression. Because it is possible that the receiver has not yet called mpiRecv even if the message has left the sending node, completion of mpiSend is not a confirmation that it has been received.

TABLE B

Call	Description
mpiSend[expr, target, comm, tag]	Sends an expression expr to a node with the ID target in the communicator world comm, waiting until that expression has left this kernel
mpiRecv [expr, target, comm, tag]	Receives an expression into expr from a node with the ID target in the communicator world comm, waiting until the expression has arrived
mpiSendRecv[sendexpr, dest, recvexpr, source, comm]	Simultaneously sends the expression sendexpr to the node with the ID target and receives an expression into recvexpr from the node with the ID source in the communicator world comm, waiting until both operations have returned.

Asynchronous MPI Calls

Asynchronous calls make it possible for the kernel to do work while communications are proceeding simultaneously. It is also possible that another node may not be able to send or receive data yet, allowing one kernel to continue working while waiting.

TABLE C

Call	Description
mpiISend[expr, target, comm, tag, req]	Sends an expression expr to a processor with the ID target in the communicator world comm, returning immediately. It can be balanced with calls to mpiTest[req] until mpiTest[req] returns True.
mpiIRecv[expr, target, comm, tag, req]	Receives an expression expr from a processor with the ID target in the communicator world comm, returning immediately. It can be balanced with calls to mpiTest[req] until mpiTest[req] returns True. The expr is not safe to access until mpiTest[req] returns True.
mpiTest[req]	Completes asynchronous behavior of mpiISend and mpiIRecv
mpWait[req]	Calls mpiTest until it returns True.
mpiWaitall[reqlist]	Calls mpiWait all on every element of reqlist
mpiWaitany[reqlist]	Calls mpiTest on each element of reqlist until one of them returns True

14

The mpiSend[] command can be called from within a kernel module 206 (for example, a Mathematica kernel). It creates a packet containing the Mathematica expression to be sent as payload and where the expression should be sent.

5 The packet itself is destined only for its local cluster node module. Once received by its local cluster node module, this packet is decoded and its payload is forwarded on to the cluster node module specified in the packet.

10 The mpiRecv[] command can also be called from within a kernel module 206. It creates a packet specifying where it expects to receive an expression and from which processor this expression is expected. Once received by its local cluster node module, this packet is decoded and its contents are stored in a message receiving queue (MRQ) 308 (FIG. 3).

15 The mpiTest[] command can be called from within a kernel module 206. It creates a packet specifying which message to test for completion, then waits for a reply expression to evaluate. Once received by the kernel module's associated cluster node module 204, this packet is decoded and its message specifier is used to search for any matching expressions listed as completed in its received message queue (RMQ) 306. If such completed expressions are found, it is sent to its local kernel module as part of the reply in mpiTest[]. The kernel module receives this reply expression and evaluates it, which updates the kernel module's variables as needed.

20 Other MPI calls are built on the fundamental calls mpiSend, mpiRecv, and mpiTest. For example, mpiBcast, a broadcast, creates instructions to send information from the broadcast processor to all the others, while the other processors perform a Recv. Similarly, high-level calls of the toolkit can be built on top of the collection of MPI calls.

Collective MPI Calls

25 In one embodiment, the MPI module 302 can include program code for implementing collective MPI calls (for example, calls that provide basic multi-node data movement across nodes). Collective MPI calls can include broadcasts, gathers, transpose, and other vector and matrix operations, for example. Collective calls can also provide commonly used mechanisms to send expressions between groups of nodes.

TABLE D

Call	Description
mpiBcast[expr, root, comm]	Performs a broadcast of expr from the root processor to all the others in the communicator world comm. An expression is expected to be supplied by the root processor, while all the others expect expr to be overwritten by the incoming expression.
mpiGather[sendexpr, recvexpr, root, comm]	50 All processors (including root) in the communicator comm send their expression in sendexpr to the root processor, which produces a list of these expressions, in the order according to comm, in recvexpr. On the processors that are not root, recvexpr is ignored.
mpiAllgather[sendexpr, recvexpr, comm]	55 All processors in the communicator comm send their expression in sendexpr, which are organized into a list of these expressions, in the order according to comm, in recvexpr on all processors in comm.
mpiScatter[sendexpr, recvexpr, root, comm]	60 Processor root partitions the list in sendexpr into equal parts (if possible) and places each piece in recvexpr on all the processors (including root) in the communicator world comm, according the order and size of comm.

US 10,333,768 B2

15

TABLE D-continued

Call	Description
mpiAlltoall[sendexpr, recvexpr, comm]	Each processor sends equal parts of the list in sendexpr to all other processors in the communicator world comm, which each collects from all other processors and organizes into the order according to comm.

In one embodiment, the MPI module **302** includes program code for implementing parallel sums and other reduction operations on data stored across many nodes. MPI module **302** can also include program code for implementing simple parallel input/output calls (for example, calls that allow cluster system **200** to load and store objects that are located on a plurality of nodes).

TABLE E

Call	Description
mpiReduce[sendexpr, recvexpr, operation, root, comm]	Performs a collective reduction operation between expressions on all processors in the communicator world comm for every element in the list in sendexpr returning the resulting list in recvexpr on the processor with the ID root.
mpiAllreduce[sendexpr, recvexpr, operation, comm]	Performs a collective reduction operation between expressions on all processors in the communicator world comm for every element in the list in sendexpr returning the resulting list in recvexpr on every processor.
mpiReduceScatter[sendexpr, recvexpr, operation, comm]	Performs a collective reduction operation between expressions on all processors in the communicator world comm for every element in the list in sendexpr, partitioning the resulting list into pieces for each processor's recvexpr.

These additional collective calls perform operations that reduce the data in parallel. The operation argument can be one of the constants below.

TABLE F

Constant	Description
mpiSum	Specifies that all the elements on different processors be added together in a reduction call
mpiMax	Specifies that the maximum of all the elements on different processors be chosen in a reduction call
mpiMin	Specifies that the minimum of all the elements on different processors be chosen in a reduction call

MPI Communicator Calls

In one embodiment, the MPI module **302** includes program code for implementing communicator world calls (for example, calls that would allow subsets of nodes to operate as if they were a sub-cluster). Communicators organize groups of nodes into user-defined subsets. The communicator values returned by mpiCommSplit[] can be used in other MPI calls instead of mpiCommWorld.

TABLE G

Call	Description
mpiCommSize[comm]	Returns the number of processors within the communicator comm
mpiCommRank[comm]	Returns the rank of this processor in the communicator comm

16

TABLE G-continued

Call	Description
mpiCommDup[comm]	Returns a duplicate communicator of the communicator comm
mpiCommSplit[comm, color, key]	Creates a new communicator into several disjoint subsets each identified by color. The sort order within each subset is first by key, second according to the ordering in the previous communicator. Processors not meant to participate in any new communicator indicates this by passing the constant mpiUndefined. The corresponding communicator is returned to each calling processor.
mpiCommMap[comm] mpiCommMap[comm, target]	Returns the mapping of the communicator comm to the processor indexed according to \$mpiCommWorld. Adding a second argument returns just the ID of the processor with the ID target in the communicator comm.
mpiCommFree[comm]	Frees the communicator comm

Other MPI Support Calls

Other calls that provide common functions include:

TABLE H

Call	Description
mpiWtime[]	Provides wall-dock time since some fixed time in the past. There is no guarantee that this time will read the same on all processors.
mpWtick[] MaxByElement[in]	Returns the time resolution of mpiWtime[] For every nth element of each list of the list in, chooses the maximum according to Max[], and returns the result as one list. Used in the mpiMax reduction operation.
MinByElement[in]	For every nth element of each list of the list in, chooses the minimum according to Min[], and returns the result as one list. Used in the mpiMin reduction operation.

2. Advanced Functions Module **304**

In one embodiment, the cluster node module **204** includes an advanced functions module **304**. The advanced functions module **304** can include program code that provides a toolkit of functions inconvenient or impractical to do with MPI instructions and calls implemented by the MPI module **302**. The advanced functions module **304** can rely at least partially on calls and instructions implemented by the MPI module **302** in the implementation of advanced functions. In one embodiment, the advanced functions module **304** includes a custom set of directives or functions. In an alternative embodiment, the advanced functions module **304** intercepts normal Mathematica language and converts it to one or more functions optimized for cluster execution. Such an embodiment can be easier for users familiar with Mathematica functions to use but can also complicate a program debugging process. Some functions implemented by the advanced functions module **304** can simplify operations difficult or complex to set up using parallel computing. Several examples of such functions that can be implemented by the advanced functions module **304** are shown below.

Built on the MPI calls, the calls that are described below provide commonly used communication patterns or parallel versions of Mathematica features. Unless otherwise specified, these are executed in the communicator mpiCommWorld, whose default is \$mpiCommWorld, but can be changed to a valid communicator at run time.

US 10,333,768 B2

17

Common Divide-and-Conquer Parallel Evaluation

In one embodiment, the advanced functions module **304** includes functions providing for basic parallelization such as, for example, routines that would perform the same operations on many data elements or inputs, stored on many nodes. These functions can be compared to parallelized for-loops and the like. The following calls address simple parallelization of common tasks. In the call descriptions, “expr” refers to an expression, and “loopspec” refers to a set of rules that determine how the expression is evaluated. In some embodiments, the advanced functions module **304** supports at least three forms of loopspec, including {var, count}, where the call iterates the variable var from 1 to the integer count; {var, start, stop}, where the call iterates the variable var every integer from start to stop; and {var, start, stop, increment}, where the call iterates the variable var from start adding increment for each iteration until var exceeds stop, allowing var to be a non-integer.

TABLE I

Call	Description
ParallelDo[expr, loopspec]	Like Do[] except that it evaluates expr across the cluster, rather than on just one processor. The rules for how expr is evaluated is specified in loopspec, like in Do[].
ParallelFunctionToList[f, count]	Evaluates the function f[i] from 1 to count, but across the cluster, and returns these results in a list. The third argument has it gather this list into the processor whose ID is root.
ParallelFunctionToList[f, count, root]	
ParallelTable[expr, loopspec]	Like Table[] except that it evaluates expr across the cluster, rather than on just one processor, returning the locally evaluated portion. The third argument has it gather this table in to the processor whose ID is root.
ParallelTable[expr, loopspec, root]	
ParallelFunction[f, inputs, root]	Like f[inputs] except that it evaluates f on a subset of inputs scattered across the cluster from processor root and gathered back to root.
ParallelNintegrate[expr, loopspec]	Like Nintegrate[] except that it evaluates a numerical integration

18

TABLE I-continued

Call	Description
ParallelNintegrate[expr, loopspec, digits]	of expr over domains partitioned into the number of processors in the cluster, then returns the sum. The third argument has each numerical integration execute with at least that many digits of precision.

Guard-Cell Management

In one embodiment, the advanced functions module **304** includes functions providing for guard-cell operations such as, for example, routines that perform nearest-neighbor communications to maintain edges of local arrays in any number of dimensions (optimized for 1-, 2-, and/or 3-D). Typically the space of a problem is divided into partitions. Often, however, neighboring edges of each partition can interact, so a “guard cell” is inserted on both edges as a substitute for the neighboring data. Thus the space a processor sees is two elements wider than the actual space for which the processor is responsible. EdgeCell helps maintain these guard cells.

TABLE J

Call	Description
EdgeCell[list]	Copies the second element of list to the last element of the left processor and the second-to-last element of list to the first element of the right processor while simultaneously receiving the same from its neighbors.

Matrix and Vector Manipulation

The advanced functions module **304** can also include functions providing for linear algebra operations such as, for example, parallelized versions of basic linear algebra on structures partitioned on many nodes. Such linear algebra operations can reorganize data as needed to perform matrix and vector multiplication or other operations such as determinants, trace, and the like. Matrices are partitioned and stored in processors across the cluster. These calls manipulate these matrices in common ways.

TABLE K

Call	Description
ParallelTranspose[matrix]	Like Transpose[] except that it transposes matrix that is in fact represented across the cluster, rather than on just one processor. It returns the portion of the transposed matrix meant for that processor.
ParallelProduct[matrix, vector]	Evaluates the product of matrix and vector, as it would on one processor, except that matrix is represented across the cluster.
ParallelDimensions[matrix]	Like Dimensions[] except that matrix is represented across the cluster, rather than on just one processor. It returns a list of each dimension.
ParallelTr[matrix]	Like Tr[] except that the matrix is represented across the cluster, rather than on just one processor. It returns the trace of this matrix.
ParallelIdentity[rank]	Like Identity[], it generates a new identity matrix, except that the matrix is represented across the cluster, rather than on just one processor. It returns the portion of the new matrix for this processor.
ParallelOuter[f, vector1, vector2]	Like Outer[f, vector1, vector2] except that the answer becomes a matrix represented across the cluster, rather than on just one processor. It returns the portion of the new matrix for this processor.
ParallelInverse[matrix]	Like Inverse[] except that the matrix is represented across the cluster, rather than on just one processor. It returns the inverse of the matrix.

US 10,333,768 B2

19

Element Management

In one embodiment, the advanced functions module **304** includes element management operations. For example, a large bin of elements or particles cut up in space across the nodes may need to migrate from node to node based on rules or criteria (such as their spatial coordinate). Such operations would migrate the data from one node to another. Besides the divide-and-conquer approach, a list of elements can also be partitioned in arbitrary ways. This is useful if elements need to be organized or sorted onto multiple processors. For example, particles of a system may drift out of the space of one processor into another, so their data would need to be redistributed periodically.

TABLE L

Call	Description
ElementManage[list, switch]	Selects which elements of list will be sent to which processors according to the function switch[] is evaluated on each element of list. If switch is a function, switch[] should return the ID of the processor that element should be sent. If switch is an integer, the call assumes that each elements is itself a list, whose first element is a number ranging from 0 to the passed argument. This call returns a list of the elements, from any processor, that is switch selected for this processor.
ElementManage[list]	Each element of list can be a list of two elements, the first being the ID of the processor where the element should be sent, while the second is arbitrary data to send. This call returns those list elements, from any and all processors, whose first element is this processors ID in a list. This call is used internally by the two-argument version of ElementManage[].

20

Fourier Transform

In one embodiment, the advanced functions module **304** includes program code for implementing large-scale parallel fast Fourier transforms (“FFTs”). For example, such functions can perform FFTs in one, two, and/or three dimensions on large amounts of data that are not stored on one node and that are instead stored on many nodes. Fourier transforms of very large arrays can be difficult to manage, not the least of which is the memory requirements. Parallelizing the Fourier transform makes it possible to make use of all the memory available on the entire cluster, making it possible to manipulate problem sizes that no one processor could possibly do alone.

TABLE M

Call	Description
ParallelFourier[list]	Like Fourier[] except that list is a two- or three-dimensional list represented across the cluster, like for matrices, above. It returns the portion of the Fourier-transformed array meant for that processor.

Parallel Disk I/O

In one embodiment, the advanced functions module **304** includes parallel disk input and output calls. For example, data may need to be read in and out of the cluster in such a way that the data is distributed across the cluster evenly. The calls in the following table enable the saving data from one or more processors to storage and the retrieval data from storage.

TABLE N

Call	Description
ParallelPut[expr, filename]	Puts expr into the file with the name filename in order on processor 0. The third argument specifies that the file be written on the processor whose ID is root. The fourth uses the communicator world comm.
ParallelPut[expr, filename, root]	
ParallelPut[expr, filename, root, comm]	
ParallelGet[filename]	Reads and returns data from the file with the name filename on processor 0 partitioned into each processor on the cluster. The second argument specifies that the file is to be read on the processor whose ID is root. The third uses the communicator world comm.
ParallelGet[filename, root]	
ParallelGet[filename, root, comm]	
ParallelBinaryPut[expr, type, filename]	Puts expr into the file with the binary format type with the name filename in order on processor 0. The fourth argument specifies that the file be written on the processor whose ID is root. The fifth uses the communicator world comm.
ParallelBinaryPut[expr, filename, root]	
ParallelBinaryPut[expr, filename, root, comm]	
ParallelBinaryGet[type, filename]	Reads and returns data in the binary format type from the file with the name filename on processor 0 partitioned into each processor on the cluster. The third argument specifies that the file is to be read on the processor whose ID is root. The fourth uses the communicator world comm.
ParallelBinaryGet[type, filename, root]	
ParallelBinaryGet[type, filename, root, comm]	
ParallelGetPerProcessor[expr, filename]	Puts expr into the file with the name filename in order on processor 0, one line per processor. The third argument specifies that the file be written on the processor whose ID is root. The fourth uses the communicator world comm.
ParallelGetPerProcessor[filename, root]	
ParallelGetPerProcessor[filename, root, comm]	

US 10,333,768 B2

21

TABLE N-continued

Call	Description
ParallelGetPerProcessor [filename]	Reads and returns data from the file with the name filename on processor 0, one line for each processor. The second argument
ParallelGetPerProcessor [filename, root]	specifies that the file is to be read on the processor whose ID is root. The third uses the communicator world comm.
ParallelGetPerProcessr [filename, root, comm]	

22

Automatic Load Balancing

Some function calls can take an inconsistent amount of processing time to complete. For example, in Mathematica, the call f[20] could in general take much longer to evaluate than f[19]. Moreover, if one or more processors within the cluster are of different speeds (for example, if some operate at a core frequency of 2.6 GHz while other operate at less than one 1 GHz), one processor may finish a task sooner than another processor.

In some embodiments, the advanced functions module 304 includes a call that can improve the operation of the computer cluster 100 in such situations. In some embodiments, the root processor assigns a small subset of the possible calls for a function to each processor on the cluster 100. Whichever processor returns its results first is assigned a second small subset of the possible calls. The root processor will continue to assign small subsets of the possible calls as results are received until an evaluation is complete. The order in which the processors finish can vary every time an expression is evaluated, but the root processor will continue assigning additional work to processors as they become available.

In one illustrative example, there are 4 processors and f[1] to f[100] to evaluate. One could implement this by assigning f[1], f[2], f[3], f[4] to each of processors 0 (the root can assign to oneself) through 3. If the f[2] result came back first, then processor 1 would be assigned f[5]. If the f[4] result is returned next, f[6] would be assigned to processor 3. The assignments continue until all results are calculated. The results are organized for output back to the user.

In alternative embodiments, the subsets of possible calls can be assigned in any order, rather than sequentially, or in batches (for example, f[1], f[5], f[9] assigned to processor 1, etc.). Also, the subsets could be organized by delegation. For example, one processor node may not necessarily be in direct control of the other processors. Instead, a large subset could be assigned to a processor, which would in turn assign subsets of its work to other processors. The result would create a hierarchy of assignments like a vast army.

TABLE O

Call	Description
LoadBalanceFunctionToList[f, count]	Evaluates the function f[i] from 1 to count, but across the cluster
LoadBalanceFunctionToList[f, count, root]	using load-balancing techniques, and returns these results in a list. The third argument has it gather this list into the processor whose ID is root.

3. Received Message Queue 306

In one embodiment, the cluster node module 204 includes a received message queue 306. The received message queue 306 includes a data structure for storing messages received from other cluster node modules. Related data pertaining to the messages received, such as whether an expression has

been completed, may also be stored in the received message queue 306. The received message queue 306 may include a queue and/or another type of data structure such as, for example, a stack, a linked list, an array, a tree, etc.

4. Message Receiving Queue 308

In one embodiment, the cluster node module 204 includes a message receiving queue 308. The message receiving queue 308 includes a data structure for storing information about the location to which an expression is expected to be sent and the processor from which the expression is expected. The message receiving queue 308 may include a queue and/or another type of data structure such as, for example, a stack, a linked list, an array, a tree, etc.

B. Cluster Configuration Module 208

Cluster configuration module 208 includes program code for initializing a plurality of cluster node modules to add cluster computing support to computer systems 110, 120, 130. U.S. Pat. No. 7,136,924, issued to Dauger (the “’924 patent”), the entirety of which is hereby incorporated by reference and made a part of this specification, discloses a method and system for parallel operation and control of computer clusters. One method generally includes obtaining one or more personal computers having an operating system with discoverable network services. In some embodiments, the method includes obtaining one or more processors or processor cores on which a kernel module can run. As described in the ’924 patent, a cluster node control and interface (CNCI) group of software applications is copied to each node. When the CNCI applications are running on a node, the cluster configuration module 208 can permit a cluster node module 204, in combination with a kernel module 206, to use the node’s processing resources to perform a parallel computation task as part of a computer cluster. The cluster configuration module 208 allows extensive automation of the cluster creation process in connection with the present disclosure.

C. User Interface Module 202

In some embodiments, computer cluster 100 includes a user interface module 202, such as, for example a Mathematica Front End or a command line interface, that includes program code for a kernel module 206 to provide graphical output, accept graphical input, and provide other methods of user communication that a graphical user interface or a command-line interface provides. To support a user interface module 202, the behavior of a cluster node module 204a is altered in some embodiments. Rather than sending output to and accepting input from the user directly, the user interface module 202 activates the cluster node module 204a to which it is connected and specifies parameters to form a connection, such as a MathLink connection, between the cluster node module 204a and the user interface module 202. The user interface module’s activation of the cluster node module 204a can initiate the execution of instructions to activate the remaining cluster node modules 204b-e on the cluster and to complete the sequence to start all kernel modules 206a-e on the cluster. Packets from the user interface

module **202**, normally intended for a kernel module **206a**, are accepted by the cluster node module **204a** as a user command. Output from the kernel module **206a** associated with the cluster node module **204a** can be forwarded back to the user interface module **202** for display to a user. Any of the cluster node modules **204a-e** can be configured to communicate with a user interface module **202**.

D. Kernel Module **206**

A kernel module **206** typically includes program code for interpreting high-level code, commands, and/or instructions supplied by a user or a script into low-level code, such as, for example, machine language or assembly language. In one embodiment, each cluster node module **204a-e** is connected to all other cluster node modules, while each kernel module **206a-e** is allocated and connected only to one cluster node module **204**. In one embodiment, there is one cluster node module-kernel module pair per processor. For example, in an embodiment of a computer cluster **100** including single-processor computer systems, each cluster node module-kernel module pair could reside on a single-processor computer. If a computer contains multiple processors or processing cores, it may contain multiple cluster node module-kernel module pairs, but the pairs can still communicate over the cluster node module's network connections.

IV. Cluster Computing Methods

In one embodiment, the computer cluster **100** includes a cluster initialization process, a method of cluster node module operation, and a cluster shut down process.

A. Cluster Initialization Process

In one embodiment, a cluster configuration module **202** initializes one or more cluster node modules **204** in order to provide cluster computing support to one or more kernel modules **206**, as shown in FIG. 4.

At **402**, cluster node modules are launched on the computer cluster **100**. In one embodiment, the cluster node module **204a** running on a first processor **112a** (for example, where the user is located) accesses the other processors **112b**, **122a-b**, **132** on the computer cluster **100** via the cluster configuration module **208** to launch cluster node modules **204b-e** onto the entire cluster. In an alternative embodiment, the cluster configuration module **208** searches for processors **112a-b**, **122a-b**, **132** connected to one another via communications network **102** and launches cluster node modules **204a-e** on each of the processors **112a-b**, **122a-b**, **132**.

The cluster node modules **204a-e** establish communication with one another at **404**. In one embodiment, each of the cluster node modules **204a-e** establish direct connections using the MPI_Init command with other cluster node modules **204a-e** launched on the computer cluster **100** by the cluster configuration module **208**.

At **406**, each cluster node module **204** attempts to connect to a kernel module **206**. In one embodiment, each instance of the cluster node modules **204a-e** locates, launches, and connects with a local kernel module via MathLink connections and/or similar connection tools, for example, built into the kernel module **206**.

At **408**, the cluster node modules **204** that are unconnected to a kernel module **206** are shut down. In one embodiment, each cluster node module **204** determines whether the local kernel module cannot be found or connected to. In one embodiment, each cluster node module **204**

reports the failure to connect to a kernel module **206** to the other cluster node modules on computer cluster **100** and quits.

Processor identification numbers are assigned to the remaining cluster node modules **204** at **410**. In one embodiment, each remaining cluster node module **204** calculates the total number of active processors (N) and determines identification numbers describing the remaining subset of active cluster node modules **204a-e** and kernel modules **206a-e**. This new set of cluster node module-kernel module pairs may be numbered 0 through N-1, for example.

Message passing support is initialized on the kernel modules **206a-e** at **412**. In one embodiment, each cluster node module **204** supplies initialization code (for example, Mathematica initialization code) to the local kernel module **206** to support message passing.

Finally, at **414**, the cluster node modules **204a-e** enter a loop to accept user entry. In one embodiment, a main loop (for example, a cluster operation loop) begins execution after the cluster node module **204a** on the first processor **112a** returns to user control while each of the other cluster node modules **204** waits for messages from all other cluster node modules **204a-e** connected to the network **102**.

The initialization process creates a structure enabling a way for the kernel modules **206a-e** to send messages to one another. In some embodiments, any kernel module can send data to and receive data from any other kernel module within the cluster when initialization is complete. The cluster node module creates an illusion that a kernel module is communicating directly with the other kernel modules. The initialization process can create a relationship among kernel modules on a computer cluster **100** such as the one shown by way of example in FIG. 2.

B. Cluster Node Module Operation

In one embodiment, a cluster node module **204** implements cluster computing support for a kernel module **206** during a main loop, as shown in FIG. 5.

At **502**, cluster node modules **204** wait for user commands or messages from other cluster node modules. In one embodiment, the cluster node module **204a** connected to the user interface module **202** waits for a user command, while the other cluster node modules **204b-e** continue checking for messages.

Once a command or message is received, the method proceeds to **504**. At **504**, the cluster node module **204a** determines whether the message received is a quit command. If a quit command is received, the cluster node module **204a** exits the loop and proceeds to a cluster node module shut down process at **505**. If the message received is not a quit command, the process continues to **506**.

At **506**, received commands are communicated to all cluster node modules **204a-e** on the computer cluster **100**. In one embodiment, when a user enters a command in the user interface module **202**, the cluster node module **204a** connected to the user interface module **202** submits the user command to all other cluster node modules **204b-e** in the computer cluster **100**. The user commands can be simple (for example, "1+1"), but can also be entire subroutines and sequences of code (such as, for example, Mathematica code), including calls to MPI from within the user interface module **202** (for example, the Mathematica Front End) to perform message passing between kernel modules **206a-e** (for example, Mathematica kernels). These include the fundamental MPI calls, which are implemented using specially identified messages between a cluster node module **204** and its local kernel module **206**.

US 10,333,768 B2

25

The message (or user command) is communicated to the kernel modules **206a-e** at **508**. In one embodiment, the cluster node module **204a** connected to the user interface module **202** submits the user command to the kernel module **206a** to which it is connected. Each of the other cluster node modules **204b-e**, after receiving the message, submits the command to the respective kernel module **206b-e** to which it is connected.

At **510**, a cluster node module **204** receives a result from a kernel module **206**. In one embodiment, once the kernel module **206** completes its evaluation, it returns the kernel module's output to the cluster node module **204** to which it is connected. Depending on the nature of the result from the kernel module, the cluster node module **204** can report the result to a local computer system or pass the result as a message to another cluster node module **204**. For example, the cluster node module **204a** running on the first processor **112a** reports the output on its local computer system **110**. For example, on the first processor **112a**, cluster node module **204a** only directly reports the output of kernel module **206a**.

Messages from other cluster node modules **204** are responded to at **512**. In one embodiment, each cluster node module (for example, the cluster node module **204a**) checks for and responds to messages from other cluster node modules **204b-e** and from the kernel module **206a** repeatedly until those are exhausted. In one embodiment, output messages from the kernel module **206** are forwarded to output on the local computer system. Messages from other cluster node modules **204** are forwarded to a received message queue **306** ("RMQ"). Data from each entry in the message receiving queue **308** ("MRQ") is matched with entries in the RMQ **306** (see, for example, description of the `mpiRecv[]` call, above). If found, data from the MRQ **308** are combined into those in the RMQ **306** and marked as "completed" (see, for example, description of the `mpiTest[]` call, above). This process provides the peer-to-peer behavior of the cluster node modules **204a-e**. Via this mechanism, code running within multiple, simultaneously running kernel modules (for example, Mathematica kernels) can interact on a pair-wise or collective basis, performing calculations, processing, or other work on a scale larger and/or faster than one kernel could have done alone. In this manner, user-entered instructions and data specifying what work will be done via user commands can be executed more quickly and/or reliably. Once responding to messages has completed, the process returns to **502**.

In some embodiments, a computer system includes software, such as an operating system, that divides memory and/or other system resources into a user space, a kernel space, an application space (for example, a portion of the user space allocated to an application program), and/or an operating system space (for example, a portion of the user space allocated to an operating system). In some embodiments, some or all of the cluster node modules **204a-e** are implemented in the application space of a computer system. In further embodiments, at least some of the cluster node modules **204a-e** are implemented in the operating system space of a computer system. For example, some cluster node modules in a computer cluster may operate in the application space while others operate in the operating system space.

In some embodiments, some or all of the functionality of the cluster node modules **204a-e** is incorporated into or integrated with the operating system. The operating system can add cluster computing functionality to application programs, for example, by implementing at least some of the methods, modules, data structures, commands, functions,

26

and processes discussed herein. Other suitable variations of the techniques described herein can be employed, as would be recognized by one skilled in the art.

In some embodiments, the operating system or components of the operating system can identify and launch the front end **202** and the kernels **206**. The operating system or its components can connect the front end **202** and kernels **206** to one another in the same manner as a cluster node module **204** would or by a variation of one of the techniques described previously. The operating system can also be responsible for maintaining the communications network **102** that connects the modules to one another. In some embodiments, the operating system implements at least some MPI-style calls, such as, for example, collective MPI-style calls. In some embodiments, the operating system includes an application programming interface (API) library of cluster subroutine calls that is exposed to application programs. Applications programs can use the API library to assist with launching and operating the computer cluster.

C. Cluster Shut Down Process

In one embodiment, a computer cluster **100** includes a procedure to shut down the system. If the operation process (or main loop) on the cluster node module **204a** connected to the user interface module **202** detects a "Quit" or "Exit" command or otherwise receives a message from the user indicating a shut down, the sequence to shut down the cluster node modules **204a-e** and the kernel modules **206a-e** is activated. In one embodiment, the cluster node module **204a** connected to the user interface module **202** sends a quit message to all other cluster node modules **204b-e**. Each cluster node module **204** forwards the quit command to its local kernel module **206**. Once its Mathematica kernel has quit, each cluster node module **204** proceeds to tear down its communication network with other cluster node modules (for example, see description of the `MPI_Finalize` command, above). At the conclusion of the process, each cluster node module **204** exits execution.

V. Example Operation

For purposes of illustration, sample scenarios are discussed in which the computer cluster system is used in operation. In these sample scenarios, examples of Mathematica code are given, and descriptions of how the code would be executed by a cluster system are provided.

Basic MPI

Fundamental data available to each node includes the node's identification number and total processor count.

```
In[1]:= { $IdProc, $NProc }
Out[1]:= { 0, 2 }
```

The first element should be unique for each processor, while the second is generally the same for all. Processor 0 can see what other values are using a collective (see below) communications call such as `mpiGather[]`.

```
In[2]:= mpiGather[{ $IdProc, $NProc }, list, 0]; list
Out[2]:= { { 0, 2 }, { 1, 2 } }
```

Peer-to-Peer MPI

The `mpiSend` and `mpiRecv` commands make possible basic message passing, but one needs to define which

US 10,333,768 B2

27

processor to target. The following defines a new variable, targetProc, so that each pair of processors will point to each other.

```
In[3]:= targetProc=If[1==Mod[$IdProc, 2],$IdProc-1,$IdProc+1]
Out[3]:= 1
```

In this example, the even processors target its “right” processor, while the odd ones point its “left.” For example, if the processors were lined up in a row and numbered in order, every even-numbered processor would pair with the processor following it in the line, and every odd-numbered processor would pair with the processor preceding it. Then a message can be sent:

```
In[4]:= If[ 1==Mod[ $IdProc , 2],mpiSend[N[Pi,22],targetProc,
mpiCommWorld,d], mpiRecv[a,targetProc,mpiCommWorld,d]]
```

The If [] statement causes the processors to evaluate different code: the odd processor sends 22 digits of Pi, while the even receives that message. Note that these MPI calls return nothing. The received message is in the variable a:

```
In[5]:= a
Out[5]:= 3.1415926535897932384626
In[6]:= Clear[a]
```

The variable a on the odd processors would have no definition. Moreover, if \$NProc is 8, processor 3 sent Pi to processor 2, processor 5 sent Pi to processor 4, and so on. These messages were not sent through processor 0, but they communicated on their own.

The mpiSend and mpiRecv commands have a letter “I” to indicate asynchronous behavior, making it possible to do other work while messages are being sent and received, or if the other processor is busy. So, the above example could be done asynchronously:

```
In[7]:= If[1==Mod[$IdProc, 2],mpiISend[N[Pi,22],targetProc,
mpiCommWorld,d,e],
mpiIRecv[a,targetProc,mpiCommWorld,d,e]]
```

The variable e has important data identifying the message, and mpiTest[e] can return True before the expressions are to be accessed. At this point, many other evaluations can be performed. Then, one can check using mpiTest when the data is needed:

```
In[29]:= mpiTest[e]
Out[29]:= True
In[30]:= a
Out[30]:= 3.1415926535897932384626
In[31]:= Clear[a,e]
```

The mpiWait[e] command could have also have been used, which does not return until mpiTest[e] returns True. The power of using these peer-to-peer calls is that it becomes possible to construct any message-passing pattern for any problem.

28

Collective MPI

In some cases, such explicit control is not required and a commonly used communication pattern is sufficient. Suppose processor 0 has an expression in b that all processors are meant to have. A broadcast MPI call would do:

```
In[8]:=mpiBcast[b, 0, mpiCommWorld]
```

The second argument specifies which processor is the “root” of this broadcast; all others have their b overwritten. To collect values from all processors, use mpiGatherD:

```
In[9]:=mpiGather[b, c, 0, mpiCommWorld]
```

The variable c of processor 0 is written with a list of all the b of all the processors in mpiCommWorld. The temporal opposite is mpiScatter:

```
In[10]:= Clear[b]; a = {2, 4, 5, 6}; mpiScatter[a, b, 0,
mpiCommWorld]; b
Out[10]:= {2, 4}
```

The mpiScatter command cuts up the variable a into even pieces (when possible) and scatters them to the processors. This is the result if \$NProc=2, but if \$NProc=4, b would only have {2}.

MPI provides reduction operations to perform simple computations mixed with messaging. Consider the following:

```
In[11]:= a = {{2 + $IdProc, 45[ ],3,{1 + $IdProc,$NProc[ ]}];
mpiReduce [a,d,mpiSum, 0,mpiCommWorld ]
In[12]:= d
Out[12]:= {{5, 90}, 6, {3, 4}}
```

The mpiSum constant indicates that variable a of every processor will be summed. In this case, \$NProc is 2, so those elements that were not identical result in odd sums, while those that were the same are even.

Most of these calls have default values if not all are specified. For example each of the following calls will have the equivalent effect as the above mpiGather[] call:

```
mpiGather[b, c, 0]
mpiGather[b, c]
c = mpiGather[b]
```

High-Level Calls

High-level calls can include convenient parallel versions of commonly used application program calls (for example, Mathematica calls). For example, ParallelTable[] is like Table[], except that the evaluations are automatically performed in a distributed manner:

```
In[13]:= ParallelTable[i,{i,100},0]
Out[13]:= {1,2,3,4,5, ..., 99,100}
```

The third argument specifies that the answers are collated back to processor 0. This is a useful, simple way to parallelize many calls to a complex function. One could define a complicated function and evaluate it over a large range of inputs:

```
In[14]:= g[x_] := Gamma[2 + 0.5*(x-1)];
ParallelTable[g[i],{i,100},0]
Out[14]:= {1, 1.32934, 2., 3.32335, 6., 11.6317, 24., 52.3428,
120., 287.885, 720}
```

US 10,333,768 B2

29

ParallelFunctionToList[] also provides a simplified way to perform this form of parallelism.

Operations with Non-Trivial Communication

Matrix Operations

In some embodiments, one or more functions can help solve matrix calculations in parallel:

```
In[15]:= a = Table[i+ 3* $IdProc + 2 j, {i, 2}, {j,4}]
Out[15]:= {{3, 5, 7, 9}, {4, 6, 8, 10}}
In[16]:= t = ParallelTranspose[a]
Out[16]:= {{3, 4, 6, 7}, {5, 6, 8, 9}}
```

Fourier Transforms

A Fourier transform of a large array can be solved faster in parallel, or made possible on a cluster because it can all be held in memory. A two-dimensional Fourier transform of the above example follows:

```
In[17]:= f = ParallelFourier[a]
Out[17]:= {{32. + 0. I, -4. - 4. I, -4., -4. + 4. I}, {-3. - 3. I, 0. + 0. I, 0., 0. + 0. I}}
```

Edge Cell Management

Many problems require interactions between partitions, but only on the edge elements. Maintaining these edges can be performed using EdgeCell[].

```
In[18]:= a = {2, 4, 5, 6, 7}+8*$IdProc
Out[18]:= {2, 4, 5, 6, 7}
In[19]:= EdgeCell[a]; a
Out[19]:= {14, 4, 5, 6, 12}
```

Element Management

In particle-based problems, items can drift through space, sometimes outside the partition of a particular processor. This can be solved with ElementManage[1:

```
In[20]:= list={{0,4},{1,3},{1,4},{0,5}}; fcn[x_]:=x[[1]]
In[21]:= ElementManage[list, fcn]
Out[21]:= {{0, 4}, {0, 5}, {0, 4}, {0, 5}}
In[21]:= ElementManage[list, 2]
Out[21]:= {{0, 4}, {0, 5}, {0, 4}, {0, 5}}
```

The second argument of ElementManage describes how to test elements of a list. The fcn identifier returns which processor is the “home” of that element. Passing an integer assumes that each element is itself a list, whose first element is a number ranging from 0 to the passed argument.

While the examples above involve Mathematica software and specific embodiments of MPI calls and cluster commands, it is recognized that these embodiments are used only to illustrate features of various embodiments of the systems and methods.

VI. Additional Embodiments

Although cluster computing techniques, modules, calls, and functions are disclosed with reference to certain embodiments, the disclosure is not intended to be limited thereby. Rather, a skilled artisan will recognize from the disclosure herein a wide number of alternatives for the exact selection of cluster calls, functions, and management systems. For example, single-node kernels can be managed using a variety of management tools and/or can be managed

30

manually by a user, as described herein. As another example, a cluster node module can contain additional calls and procedures, including calls and procedures unrelated to cluster computing, that are not disclosed herein.

Other embodiments will be apparent to those of ordinary skill in the art from the disclosure herein. Moreover, the described embodiments have been presented by way of example only, and are not intended to limit the scope of the disclosure. Indeed, the novel methods and systems described herein can be embodied in a variety of other forms without departing from the spirit thereof. Accordingly, other combinations, omissions, substitutions and modifications will be apparent to the skilled artisan in view of the disclosure herein. Thus, the present disclosure is not intended to be limited by the disclosed embodiments, but is to be defined by reference to the appended claims. The accompanying claims and their equivalents are intended to cover forms or modifications as would fall within the scope and spirit of the inventions.

What is claimed is:

1. A computer cluster comprising:

a plurality of nodes, wherein each of the plurality of nodes comprises a hardware processor, wherein one or more of the nodes are configured to receive a command to start a cluster initialization process for the computer cluster, and wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, is capable of causing the hardware processor to evaluate mathematical expressions; and a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture;

wherein the plurality of nodes comprises:

a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and

a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of the first mathematical expression evaluation to a third node;

wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;

wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface;

wherein one or more of the nodes are configured to:

accept user instructions;

after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and

US 10,333,768 B2

31

after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to one or more single-node kernels.

2. The computer cluster of claim 1, wherein the single-node kernel comprises a Mathematica kernel.

3. The computer cluster of claim 1, wherein the mechanism comprises a message passing interface.

4. The computer cluster of claim 1, wherein each of the nodes comprises one or more cluster node modules.

5. The computer cluster of claim 1, wherein each single-node kernel is stored in a non-transitory computer-readable medium and configured to accept and execute a request.

6. The computer cluster of claim 5, wherein each of the nodes comprises one or more cluster node modules, wherein each of the cluster node modules comprises instructions stored in a non-transitory computer-readable medium, and wherein the instructions, when executed by the hardware processor, cause the cluster node module to communicate with the single-node kernel and with one or more other cluster node modules.

7. The computer cluster of claim 6, wherein the plurality of cluster node modules act as a cluster.

8. The computer cluster of claim 6, wherein the plurality of cluster node modules communicate with one another to act as a cluster.

9. The computer cluster of claim 8, wherein the computer cluster includes the user interface.

10. The computer cluster of claim 9, wherein each cluster node module accepts instructions from the user interface and interprets one or more of the instructions.

11. The computer cluster of claim 1, wherein the cluster initialization process comprises establishing communication among two or more of the nodes.

12. The computer cluster of claim 1, wherein the cluster initialization process comprises configuring access by one or more of the nodes to a computer-readable medium comprising program code for the single-node kernel.

13. The computer cluster of claim 12, wherein the cluster initialization process comprises establishing message-passing support among the nodes in the cluster.

14. The computer cluster of claim 12, wherein the cluster initialization process comprises launching cluster node modules by the cluster.

15. The computer cluster of claim 14, wherein the cluster initialization process comprises assigning a processor identification number to each of the cluster node modules.

16. The computer cluster of claim 1, wherein the cluster initialization process comprises:

launching cluster node modules by the cluster; and
after launching the cluster node modules, configuring access by one or more of the nodes to a non-transitory computer-readable medium comprising program code for the single-node kernel.

17. The computer cluster of claim 1, wherein the cluster initialization process comprises:

launching cluster node modules by the cluster;
after launching the cluster node modules, establishing communication among two or more of the nodes; and
after establishing communication among two or more of the nodes, assigning a processor identification number to each of the cluster node modules.

18. The computer cluster of claim 1, wherein one or more of the nodes are configured to communicate at least some of the user instructions.

32

19. The computer cluster of claim 18, wherein one or more of the nodes are configured to communicate at least some of the user instructions to one or more single-node kernels.

20. The computer cluster of claim 1, wherein one or more of the nodes are configured to accept user instructions via one or more of the nodes.

21. The computer cluster of claim 20, wherein one or more of the nodes are configured to communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other.

22. The computer cluster of claim 1, wherein one or more of the nodes are configured to transmit at least some of the user instructions that originate from a user interface.

23. The computer cluster of claim 1, wherein one or more of the nodes are configured to parallelize at least some of the user instructions before communicating at least some of the user instructions to one or more single-node kernels.

24. The computer cluster of claim 1, wherein one or more of the nodes are configured to accept user instructions before communicating at least some of the user instructions to one or more single-node kernels.

25. The computer cluster of claim 1, wherein the plurality of nodes are configured to communicate with one another to interpret and translate commands for execution by a plurality of single-node kernels.

26. A computer cluster comprising:

a plurality of nodes, wherein one or more of the nodes are configured to receive:

a command to start a cluster initialization process for the computer cluster, wherein the cluster initialization process comprises establishing communication among two or more of the nodes; and
an instruction from a user interface or a script;

and

a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using asynchronous calls;

wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, is capable of causing a hardware processor to evaluate mathematical expressions;

wherein the plurality of nodes comprises:

a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and

a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of mathematical expression evaluation to a third node; wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;

US 10,333,768 B2

33

wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface or the script;

wherein one or more of the nodes are configured to:

- accept user instructions;
- after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and
- after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to one or more single-node kernels.

27. The computer cluster of claim 26, wherein the asynchronous calls comprise a first command to create a first packet containing:

- an expression to be sent as payload; and
 - a target node where the expression should be sent;
- wherein the first command is configured to be called from within a single-node kernel;
- wherein the single-node kernel is configured to send the first packet to a local cluster node module; and
- wherein the local cluster node module is configured to forward the expression to the target node.

28. The computer cluster of claim 27, wherein the asynchronous calls comprise a second command to create a second packet containing:

- a location where the expression is expected to be received; and
 - a sending node from which the expression is expected to be received;
- wherein the second command is configured to be called from within a single-node kernel;
- wherein the single-node kernel is configured to send the second packet to a local cluster node module; and
- wherein the local cluster node module is configured to store the second packet contents in a message receiving queue.

29. A computer cluster comprising:

a plurality of nodes, wherein one or more of the nodes are configured to receive:

- a command to start a cluster initialization process for the computer cluster, wherein the cluster initialization process comprises establishing communication among two or more of the nodes; and
- an instruction from a user interface or a script;

and

a mechanism for the nodes to communicate results of mathematical expression evaluation with each other;

wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, is capable of causing a hardware processor to evaluate mathematical expressions;

wherein the plurality of nodes comprises:

- a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and

- a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of mathematical expression evaluation to a third node;

34

wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;

and

wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface or the script;

wherein one or more of the nodes are configured to:

- accept user instructions;
- after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and
- after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to one or more single-node kernels.

30. The computer cluster of claim 4, wherein each of the plurality of nodes implements asynchronous calls that enable the single-node kernel to perform computation tasks while the cluster node modules are simultaneously communicating with one another.

31. The computer cluster of claim 4, wherein intercommunication among the plurality of single-node kernels during thread execution is enabled by the plurality of cluster node modules, and wherein the computer cluster is configured to permit exchange of information between nodes during the course of a parallel computation.

32. The computer cluster of claim 4, wherein each of the single-node kernels are configured to call a send command involving creating a packet containing:

- an expression to be sent as payload; and
 - where the expression should be sent;
- wherein, upon reception of the send command by a local cluster node module associated with the single-node kernel, the local cluster node module is configured to decode the packet and forward a payload to the cluster node module specified in the packet;

wherein each of the single-node kernels is configured to call a receipt command involving creating a packet specifying which message to test for completion and to then wait for a reply expression to evaluate;

wherein, upon reception of the receipt command by the local cluster node module associated with the single-node kernel, the local cluster node module is configured to decode the packet and use a message specifier to search for any matching expressions listed as completed in a received message queue;

wherein, upon determining that such a completed expression is found, the local cluster node module is configured to send the completed expression to a local single-node kernel associated with the cluster node module in response to the receipt command, and

wherein each of the single-node kernels is configured to receive the completed expression and to update variables used by the single-node kernel during evaluation of expressions.

33. The computer cluster of claim 1, wherein the plurality of nodes are configured to permit exchange of information between nodes during the course of parallel computation.

34. The computer cluster of claim 1, wherein each of the plurality of nodes comprises instructions executable by the

US 10,333,768 B2

35

hardware processor and configured to implement asynchronous behavior, wherein the instructions comprise:

- a first instruction to asynchronously send a payload to another node;
- a second instruction to asynchronously receive a payload from another node; and
- a third instruction to search for a payload matching a message specifier.

35. A computer cluster node for evaluating expressions in parallel with other computer cluster nodes, the computer cluster node comprising:

- a hardware processor configured to access one or more non-transitory memory devices comprising program code for a single-node kernel that, when executed, causes the hardware processor to interpret user instructions, to evaluate mathematical expressions, and to produce results of mathematical expression evaluation, wherein the hardware processor comprises multiple processor cores;
- a user connection interface configured to receive a command to start a cluster initialization process for a computer cluster;
- a mechanism to communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture; and
- program code that, when executed, is capable of causing the hardware processor to:

receive calls from a second node comprising a second hardware processor configured to access a second memory comprising program code for a user interface and program code for a second single-node kernel, the second single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; execute, using the hardware processor, at least a first mathematical expression evaluation; and communicate a result of the first mathematical expression evaluation to a third node comprising a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the computer cluster node, execute at least a

36

second mathematical expression evaluation using the result of the first mathematical expression evaluation, and communicate a result of the second mathematical expression evaluation to the first node;

- wherein the user connection interface is configured to return at least one result of mathematical expression evaluation to a user interface or a script; and
- wherein the computer cluster node is configured to:

- accept user instructions;
- after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and
- after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to the single-node kernel.

36. The computer cluster node of claim 35, wherein the one or more non-transitory memory devices comprise program code for performing a parallel fast Fourier transform, wherein the program code causes the hardware processor to evaluate a command to perform a Fourier transform on an array comprising a first data portion that is stored on the computer cluster node and a second data portion that is not stored on the computer cluster node.

37. The computer cluster node of claim 35, wherein the computer cluster node is configured to permit exchange of information with other computer cluster nodes during the course of parallel computation.

38. The computer cluster node of claim 35, wherein the computer cluster node comprises instructions executable by the hardware processor and configured to implement asynchronous behavior, wherein the instructions comprise:

- a first instruction to asynchronously send a payload to another node;
- a second instruction to asynchronously receive a payload from another node; and
- a third instruction to search for a payload matching a message specifier.

39. The computer cluster node of claim 35, wherein the hardware processor comprises a special purpose microprocessor.

* * * * *

EXHIBIT C

ZTANN.002P1C3

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor	:	Zvi Tannenbaum
App. No.	:	14/181,112
Filed	:	February 14, 2014
For	:	CLUSTER COMPUTING
Examiner	:	Not yet assigned
Art Unit	:	2447
Conf. No.	:	6874

PRELIMINARY AMENDMENT

Mail Stop Amendment

Commissioner for Patents

P.O. Box 1450

Alexandria, VA 22313-1450

Dear Sir:

Preliminary to examination on the merits, please amend the above-identified application as follows:

Amendments to the Claims are reflected in the listing of claims which begins on page 2.

Remarks begin on page 3.

Application No.: 14/181,112
Filing Date: February 14, 2014

AMENDMENTS TO THE CLAIMS

1. (Canceled)
2. (New) A computer cluster comprising:
a plurality of nodes, wherein one or more of the nodes receives a command to start a cluster initialization process for the computer cluster, and wherein each of the nodes is configured to access a computer-readable medium comprising program code for a single-node kernel module configured to interpret user instructions; and
a mechanism for the nodes to communicate with each other.
3. (New) The computer cluster of claim 2, wherein the single-node kernel module comprises a Mathematica kernel.
4. (New) The computer cluster of claim 2, wherein the mechanism comprises a message passing interface.
5. (New) The computer cluster of claim 2, wherein each of the nodes comprises one or more cluster node modules.
6. (New) The computer cluster of claim 2, wherein each single-node kernel module is stored in a computer-readable medium and configured to accept and execute a request.
7. (New) The computer cluster of claim 6, wherein each of the nodes comprises one or more cluster node modules, and each of the cluster node modules is stored in a computer-readable medium and configured to communicate with the single-node kernel and with one or more other cluster node modules.
8. (New) The computer cluster of claim 7, wherein the plurality of cluster node modules act as a cluster.
9. (New) The computer cluster of claim 7, wherein the plurality of cluster node modules communicate with one another to act as a cluster.
10. (New) The computer cluster of claim 9, wherein the computer cluster includes a user interface.
11. (New) The computer cluster of claim 10, wherein each cluster node module accepts instructions from the user interface and interprets one or more of the instructions.

Application No.: 14/181,112
Filing Date: February 14, 2014

REMARKS

Amendments

Prior to examination on the merits, please amend the above-identified application as set forth above. Applicant submits that no new matter is added by these amendments.

No Disclaimers or Disavowals

Although the present communication may include alterations to the application or claims, or characterizations of claim scope or referenced art, Applicant is not conceding in this application that previously pending claims are not patentable over the cited references. Rather, any alterations or characterizations are being made to facilitate expeditious prosecution of this application. Applicant reserves the right to pursue at a later date any previously pending or other broader or narrower claims that capture any subject matter supported by the present disclosure, including subject matter found to be specifically disclaimed herein or by any prior prosecution. Accordingly, reviewers of this or any parent, child or related prosecution history shall not reasonably infer that Applicant has made any disclaimers or disavowals of any subject matter supported by the present application.

Please charge any additional fees, including any fees for additional extension of time, or credit overpayment to Deposit Account No. 11-1410.

Respectfully submitted,

KNOBBE, MARTENS, OLSON & BEAR, LLP

Dated: September 8, 2014

By: /Lance D. Smemoe/

Lance D. Smemoe
Registration No. 66,152
Attorney of Record
Customer No. 20995
(949) 760-0404

18394794



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
 Address: COMMISSIONER FOR PATENTS
 P.O. Box 1450
 Alexandria, Virginia 22313-1450
 www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
14/181,112	02/14/2014	Zvi Tannenbaum	ZTANN.002P1C3	6874

20995	7590	07/20/2016
KNOBBE MARTENS OLSON & BEAR LLP 2040 MAIN STREET FOURTEENTH FLOOR IRVINE, CA 92614		

EXAMINER	
ASRES, HERMON	

ART UNIT	PAPER NUMBER
2449	

NOTIFICATION DATE	DELIVERY MODE
07/20/2016	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

jayna.cartee@knobbe.com
 efiling@knobbe.com

Office Action Summary**Application No.**
14/181,112**Applicant(s)**
TANNENBAUM ET AL.**Examiner**
HERMON ASRES**Art Unit**
2449**AIA (First Inventor to File)
Status**
No**-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --****Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTHS FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 02/14/2014.
☐ A declaration(s)/affidavit(s) under **37 CFR 1.130(b)** was/were filed on _____.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ An election was made by the applicant in response to a restriction requirement set forth during the interview on _____; the restriction requirement and election have been incorporated into this action.
- 4) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims*

- 5) ☒ Claim(s) 2-11 is/are pending in the application.
5a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 6) ☐ Claim(s) _____ is/are allowed.
- 7) ☒ Claim(s) 2-11 is/are rejected.
- 8) ☐ Claim(s) _____ is/are objected to.
- 9) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

* If any claims have been determined allowable, you may be eligible to benefit from the **Patent Prosecution Highway** program at a participating intellectual property office for the corresponding application. For more information, please see http://www.uspto.gov/patents/init_events/pph/index.jsp or send an inquiry to PPHfeedback@uspto.gov.

Application Papers

- 10) ☐ The specification is objected to by the Examiner.
- 11) ☐ The drawing(s) filed on _____ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

Certified copies:

- a) ☐ All b) ☐ Some** c) ☐ None of the:
1. ☐ Certified copies of the priority documents have been received.
 2. ☐ Certified copies of the priority documents have been received in Application No. _____.
 3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

** See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- 1) ☒ Notice of References Cited (PTO-892)
- 2) ☒ Information Disclosure Statement(s) (PTO/SB/08a and/or PTO/SB/08b)
Paper No(s)/Mail Date 09/10/2014, 11/18/2015
- 3) ☐ Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____
- 4) ☐ Other: _____

Application/Control Number: 14/181,112
Art Unit: 2449

Page 2

DETAILED ACTION

Claims 2-11 have been examined and are rejected.

Information Disclosure Statement

The information disclosure statement (IDS) submitted on 11/18/2015, and 09/10/2014 is in compliance with the provisions of 37 CFR 1.97. Accordingly, the information disclosure statement is being considered by the examiner.

Double Patenting

The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the “right to exclude” granted by a patent and to prevent possible harassment by multiple assignees. A nonstatutory double patenting rejection is appropriate where the claims at issue are not identical, but at least one examined application claim is not patentably distinct from the reference claim(s) because the examined application claim is either anticipated by, or would have been obvious over, the reference claim(s). See, e.g., *In re Berg*, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

Application/Control Number: 14/181,112

Page 3

Art Unit: 2449

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the reference application or patent either is shown to be commonly owned with this application, or claims an invention made as a result of activities undertaken within the scope of a joint research agreement. A terminal disclaimer must be signed in compliance with 37 CFR 1.321(b).

The USPTO internet Web site contains terminal disclaimer forms which may be used. Please visit <http://www.uspto.gov/forms/>. The filing date of the application will determine what form should be used. A web-based eTerminal Disclaimer may be filled out completely online using web-screens. An eTerminal Disclaimer that meets all requirements is auto-processed and approved immediately upon submission. For more information about eTerminal Disclaimers, refer to <http://www.uspto.gov/patents/process/file/efs/guidance/eTD-info-I.jsp>.

Claim 2 is rejected on the ground of nonstatutory double patenting as being unpatentable over claims 1 and 8 of U.S. Patent No. 8,676,877. Although the conflicting claims are not identical, they are not patentably distinct from each other because:

Instant Application		US Patent 8,676,877	
Claim	Limitation	Claim	Limitation
2	A computer cluster comprising:	1 and 8	A system for performing an instruction received from a front end by executing commands on

Application/Control Number: 14/181,112

Page 4

Art Unit: 2449

	<p>a plurality of nodes, wherein one or more of the nodes receives a command to start a cluster initialization process for the computer cluster, and wherein each of the nodes is configured to access a computer-readable medium comprising program code for a single-node kernel module configured to interpret user instructions; and a mechanism for the nodes to communicate with each other.</p>	<p>one or more special purpose microprocessors, the system comprising:</p> <p>a plurality of nodes, wherein each node is configured to access a computer-readable memory system comprising program code for a single-node kernel module,wherein each cluster node module is stored in a computer-readable memory system and configured to communicate with a single-node kernel and with one or more other cluster node modules, to accept instructions, and to interpret at least some of the instructions such that the plurality of cluster node modules communicate with one another in order to act as a cluster in executing commands using one or more hardware processors</p>
--	---	---

Claim Rejections - 35 USC § 112

The following is a quotation of 35 U.S.C. 112 (pre-AIA), second paragraph:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

The following is a quotation of pre-AIA 35 U.S.C. 112, sixth paragraph:

An element in a claim for a combination may be expressed as a means or step for performing a specified function without the recital of structure, material, or acts in support thereof, and such claim shall be construed to cover the corresponding structure, material, or acts described in the specification and equivalents thereof.

Claim 2, 6 and 7 is rejected under 35 U.S.C. 112 second paragraph as being indefinite for failing to particularly point out and distinctly claim the subject matter which the inventor(s) or applicant regards as the invention.

Application/Control Number: 14/181,112
 Art Unit: 2449

Page 5

As per claim 2, 6 and 7 “**module configured to**” is a limitations that invokes 35 U.S.C. 112, sixth paragraph. The written disclosure fails to disclose the corresponding structure, material, or acts for the claimed function. Applicant's specification is devoid of sufficient disclosure of structure for these terms as required by 112 second paragraph.

Rationale for invoking §112 6¶

Examiners will apply § 112, ¶ 6 to a claim limitation that meets the following conditions:

- (1) The claim limitation uses the phrase “means for” or “step for” or a non-structural term that does not have a structural modifier;
 - (2) The phrase “means for” or “step for” or the non-structural term recited in the claim is modified by functional language; and
 - (3) The phrase “means for” or “step for” or the non-structural term recited in the claim is not modified by sufficient structure, material, or acts for achieving the specified function.
- This modifies the 3-prong analysis in MPEP § 2181, which will be revised in due course. See Supplemental Examination, 76 FR at 7167.

“When the claim limitation does not use the phrase “means for” or “step for,” examiners should determine whether the claim limitation uses a nonstructural term (a term that is simply a substitute for the term “means for”). Examiners will apply § 112, ¶6 to a claim limitation that uses a nonstructural term associated with functional language, unless the nonstructural term is (1) preceded by a structural modifier, defined in the specification as a particular structure or known by one skilled in the art, that denotes the type of structural device (e.g., “filters”), or (2) modified by sufficient

Application/Control Number: 14/181,112

Page 6

Art Unit: 2449

structure or material for achieving the claimed function. The following is a list of non-structural terms that may invoke § 112, ¶6: “mechanism for,” “module for,” “device for,” “unit for,” “component for,” “element for,” “member for,” “apparatus for,” “machine for,” or “system for.” This list is not exhaustive, and other non-structural terms may invoke § 112, ¶6.” See *id.*

Applicant is required to:

- (a) Amend the claim so that the claim limitation will no longer be a means (or step, or non-structure terms) plus function limitation under 35 U.S.C. 112, sixth paragraph; or
- (b) Amend the written description of the specification such that it expressly recites what structure, material, or acts perform the claimed function without introducing any new matter (35 U.S.C. 132(a)).

If applicant is of the opinion that the written description of the specification already implicitly or inherently discloses the corresponding structure, material, or acts so that one of ordinary skill in the art would recognize what structure, material, or acts perform the claimed function, applicant is required to clarify the record by either:

- (a) Amending the written description of the specification such that it expressly recites the corresponding structure, material, or acts for performing the claimed function and clearly links or associates the structure, material, or acts to the claimed function, without introducing any new matter (35 U.S.C. 132(a)); or

Application/Control Number: 14/181,112

Page 7

Art Unit: 2449

(b) Stating on the record what the corresponding structure, material, or acts, which are implicitly or inherently set forth in the written description of the specification, perform the claimed function. For more information, see 37 CFR 1.75(d) and MPEP §§ 608.01(o) and 2181.

Claim Rejections - 35 USC § 101

35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claims 2-11 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. The claim(s) do not fall within at least one of the four categories of patent eligible subject matter because the claimed invention is directed to a judicial exception (i.e., a law of nature, a natural phenomenon, or an abstract idea) without significantly more. Claim(s) 2-11 are directed to computer cluster for “***a plurality of nodes wherein one or more of the nodes receives a command to start a cluster initialization process and a single-node kernel to interpret user instructions***” which under the Mayo test can be viewed as merely insignificant extrasolution activity to the judicial exception, e.g., mere data gathering in conjunction with a law of nature or abstract idea” (see Mayo, 132 S. Ct. 1297-1301). The claims do not recite additional elements that are sufficient to amount to significantly more than the judicial exception as the claims do not recite an improvement to another technology or technical field, an improvement to the functioning of a computer itself, or meaningful

Application/Control Number: 14/181,112

Page 8

Art Unit: 2449

limitations beyond generally linking the use of an abstract idea to a particular technological environment. The “**computer cluster**”, “**nodes**”, and “**single-node kernel**” are recited as performing generic computer functions routinely used in computer applications. Generic computer components recited as performing generic computer functions that are well-understood, routine, and conventional activities amount to no more than implementing the abstract idea with a computerized system.

Claims not specifically mentioned are rejected by virtue of dependency and because they do not obviate the above-recited deficiencies.

Limitation that may be enough to qualify as "significantly more"

1. Improvements to another technology /technical fields;34
2. Improvements to the functioning of the computer itself;35
3. Applying the judicial exception with, or by user of a particular machine; 36
4. Effecting a transformation or reduction of a particular article to a different state or thing; 37
5. Adding a specific limitation other than what is well-understood, routine and conventional in the field, or adding unconventional steps that confine the claim to a particular useful application.38
6. Other meaningful limitations beyond generally linking the use of an abstract idea to a particular technological environment.39

Application/Control Number: 14/181,112
 Art Unit: 2449

Page 9

34 (Id., slip op. at 15: e.g., a mathematical formula applied in a specific rubber molding process (citing *Diamond v. Diehr*, 450 U.S. 175,177-178 (1981)). In this case, the application was for a "[process] for molding raw, uncured synthetic rubber into cured precision products." The process of curing synthetic rubber depends on a number of factors including time, temperature and thickness of the mold. Using the Arrhenius equation it is possible to calculate when to open the press and to remove the cured, molded rubber. The problem was that there was, at the time the invention was made, no disclosed way to obtain an accurate measure of the temperature without opening the press. The invention solved this problem by using embedded thermocouples to constantly check the temperature, and then feeding the measured values into a computer. The computer then used the Arrhenius equation to calculate when sufficient energy had been absorbed so that the molding machine should open the press.)

35 (Id., at 2359 Alice (using a computer to obtain data, adjust account balances, and issue automated instructions - is "[p]urely 'conventional. *Mayo*, 566 U. S., at __. They do not, for example, purport to improve the functioning of the computer itself or effect an improvement in any other technology or technical field, see *Alice Corp. v. CIS Banks Int'l*. 110 USPQ2d 1976 (U.S. 2014) "[o]n their face, they are drawn to the concept of intermediated settlement, i.e., the use of a third party to mitigate settlement risk. Like the risk hedging in *Bilski*, the concept of intermediated settlement is " 'a fundamental economic practice long prevalent in our system of commerce," *ibid.*, and the use of a third-party intermediary (or "clearing house") is a building block of the

Application/Control Number: 14/181,112

Page 10

Art Unit: 2449

modern economy. Thus, intermediated settlement, like hedging, is an “abstract idea” beyond §101’s scope, pp. 7-10.))

36 (Bilski, 130 S. Ct. at 3227 (“The Court’s precedents establish that the machine-or-transformation test is a useful and important clue, and investigative tool, for determining whether some claimed inventions are processes under §101.”))

37 (Diehr, 450 U.S. at 184 (“That respondents’ claims [to a specific rubber molding process] involve the transformation of an article, in this case raw, uncured synthetic rubber, into a different state or thing cannot be disputed. The respondent’s claims describe in detail a step-by-step method for accomplishing such, beginning with the loading of a mold with raw, uncured rubber and ending with the eventual opening of the press at the conclusion of the cure.”). See also Benson, 409 U.S. at 70 (“Transformation and reduction of an article ‘to a different state or thing’ is the clue to the patentability of a process claim that does not include particular machines. So it is that a patent in the process of ‘manufacturing fat acids and glycerine from fatty bodies by the action of water at a high temperature and pressure’ was sustained in *Tilghman*, 102 U.S. at 721”).

38 (Mayo, 132 S. Ct. at 1299, 1302 (claim ineligible because the recited “instructions add nothing specific to the laws of nature other than what is well-understood, routine, conventional activity, previously engaged in by those in the field,” which was “[u]nlike, say, a typical patent on a new drug or a new way of using an existing drug”)).

Application/Control Number: 14/181,112

Page 11

Art Unit: 2449

39 (Alice Corp., 134 S. Ct. at 2360 (As to its system claims, petitioner emphasizes that those claims recite “specific hardware” configured to perform “specific computerized functions.” Brief for Petitioner 53. But what petitioner characterizes as specific hardware - a “data processing system” with a “communications controller” and “data storage unit,” for example, see App. 954,958, 1257 - is purely functional and generic. Nearly every computer will include a “communications controller” and “data storage unit” capable of performing the basic calculation, storage, and transmission functions required by the method claims. As a result, none of the hardware recited “offers a meaningful limitation beyond generally linking ‘the use of the [method] to a particular technological environment,’ that is, implementation via computers” (citing *Bilski v. Kappos* 561 U.S. at 610, 611));

Limitation that were not found to be enough to qualify as “significantly more”

1. Adding the words “encoding” (or an equivalent) with an abstract idea, or mere instructions to implement an abstract idea on a computer.⁴⁰

2. Simply appending well-understood, routine and conventional activities previously known to the industry, specified at a high level of generality, to the judicial exception, e.g., a claim to an abstract idea requiring no more than a generic computer to perform generic computer functions that are well understood, routine and conventional activities previously known to the industry;⁴¹

Application/Control Number: 14/181,112
Art Unit: 2449

Page 12

3. Adding insignificant extra-solution activity to the judicial exception, e.g., mere data gathering in conjunction with a law of nature or abstract idea; 42or

4. Generally linking the use of the judicial exception to a particular technological environment or field of use.⁴³

40 (Id., at 2358 (simply implementing a mathematical principle on a physical machine, namely a computer[i]s not a patentable application of that principle.” citing Mayo, 132 S. Ct. at 1301))

41 (Id., at 2359 (using a computer to obtain data, adjust account balances, and issue automated instructions); Mayo, 132 S. Ct. at 1300 (telling a doctor to measure metabolite levels in the blood using any known process).

42 (Mayo, 132 S. Ct. at 1297-98 (measuring metabolites of a drug administered to a patient); Flook, 437 U.S. at 589-90 (1978) (adjusting an alarm limit variable to a figure computed according to a mathematical formula)

43 (Mayo, 132 S. Ct. at 1300-01 (citing Bilski, 130 S.Ct. 3223-24 (limiting hedging to use in commodities and energy markets); Flook, 437 U.S. at 589-90.

*a copy of the federal register can be found at

<http://www.gpo.gov/fdsys/pkg/FR-2014-12-16/pdf/2014-29414.pdf>

Claims 2-11 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

Application/Control Number: 14/181,112
 Art Unit: 2449

Page 13

The claims recite “***a computer-readable medium***”. The specification is silent regarding the meaning of “***a computer-readable medium***”. This can include signal per se. Thus, applying the broadest reasonable interpretation in light of the specification and taking into account the meaning of the words in their ordinary usage they would be understood by one of ordinary skill in the art (MPEP §2111), the claim as a whole covers both transitory and not transitory media. A transitory media does not fall into any of the 4 categories of invention (Process, Machine, Manufacture, or composition of matter). Therefore claim 1 is rejected under 35 U.S.C. 101 for being directed towards non-statutory subject matter. The applicant is respectfully suggested to amend the claims to overcome the 35 USC § 101 rejection.

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of pre-AIA 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

Claims 2, and 4-11 are rejected under 35 U.S.C. 102(b) as being anticipated by Block et al. (US PGPub 2005/0021751).

As per claim 2, Block teaches a computer cluster (Block,cluster data port services within a cluster infrastructure. See paragraph [0025]) comprising:

Application/Control Number: 14/181,112

Page 14

Art Unit: 2449

a plurality of nodes, wherein one or more of the nodes receives a command to start a cluster initialization process for the computer cluster, and wherein each of the nodes is configured to access a computer-readable medium comprising program code for a single-node kernel module (Block,multiple logical or TCP connections to a designated target node with designated backup target nodes, which appears to the user (i.e., an operating system or **kernel component**) *like a single data port or data pipe*. See paragraph [0044]) configured to interpret user instructions; and a mechanism for the nodes to communicate with each other. (Block, establishment of multiple network connections between a source node, a target node and one or more backup nodes in such a manner that a cluster data port is effectively utilized as single data port. See paragraph [0025]).

As per claim 4, Block teaches wherein the mechanism comprises a message passing interface. (Block, a single data port instance may also support target to source message sends. See paragraph [0050]).

As per claim 5, Block teaches wherein each of the nodes comprises one or more cluster node modules. (Block, cluster data port services within a cluster infrastructure to provide reliable and efficient communications between nodes. See paragraph [0010]).

As per claim 6, Block teaches wherein each single-node kernel module is stored in a computer-readable medium and configured to accept and execute a request.

Application/Control Number: 14/181,112

Page 15

Art Unit: 2449

(Block, The routines executed to implementoperating system or a specific application, component, program, object, module or sequence of instructions, will also be referred to herein as "computer program code," or simply "program code." See paragraph [0039], [0049]).

As per claim 7, Block teaches wherein each of the nodes comprises one or more cluster node modules, and each of the cluster node modules is stored in a computer-readable medium and configured to communicate with the single-node kernel and with one or more other cluster node modules. (Block,multiple logical or TCP connections to a designated target node with designated backup target nodes, which appears to the user (i.e., an operating system or **kernel component**) **like a single data port or data pipe**. See paragraph [0044], [0039]).

As per claim 8, Block teaches wherein the plurality of cluster node modules act as a cluster. (Block, utilize cluster data port services within a cluster infrastructure to provide reliable and efficient communications between nodes in a clustered computer system. See paragraph [0010]).

As per claim 9, Block teaches wherein the plurality of cluster node modules communicate with one another to act as a cluster. (Block, Nodes 12, 14 and 16 are typically coupled together via a clustering network 18. See paragraph [0028]).

Application/Control Number: 14/181,112
Art Unit: 2449

Page 16

As per claim 10, Block teaches wherein the computer cluster includes a user interface. (Block, Cluster manager application 60 that provides the user interface whereby a user such as a systems administrator can manage clustering operations in the system. See paragraph [0037]).

As per claim 11, Block teaches wherein each cluster node module accepts instructions from the user interface and interprets one or more of the instructions. (Block, Cluster manager application 60 that provides the user interface whereby a user such as a systems administrator can manage clustering operations in the system. See paragraph [0037], [0039]).

Claim Rejections - 35 USC § 103

The following is a quotation of pre-AIA 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

Claim 3 is rejected under 35 U.S.C. 103(a) as being unpatentable over Block et al. (US PGPub 2005/0021751) in view of Gray (USPGPub 2007/0073705).

As per claim 3, Block teaches the computer cluster of claim 2 but doesn't teach a Mathematica kernel. In analogous art Gray teaches a Mathematica kernel. (Gray, The front end may include an interactive document referred to as a notebook similar to those

Application/Control Number: 14/181,112

Page 17

Art Unit: 2449

often used with MATHEMATICA.RTM. software systems. A notebook may include input to be sent to the kernel 104 and output received from the kernel, as well as text, graphics, palettes, etc. see paragraph [0031]).

Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to take the teaching of Gray and apply them on the teaching of Block because the kernel and the front end may be implemented on a same computing system or on different computing systems that are communicatively coupled to one another. (Gray, see paragraph [0030]).

Conclusion

The prior art made of record and not relied upon is considered pertinent to applicant's disclosure. This includes:

U.S. PGPub (2007/0094532), which describes Kernel Debugging in a cluster computing system.

U.S. Patent (6,968,359), which describes merge protocol for cluster computer system.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to HERMON ASRES whose telephone number is (571)272-4257. The examiner can normally be reached on Monday - Friday 7:30 am to 5:00 pm est.

Application/Control Number: 14/181,112
Art Unit: 2449

Page 18

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, VIVEK SRIVASTAVA can be reached on 571-272-7304. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/HERMON ASRES/
Examiner, Art Unit 2449

/VIVEK SRIVASTAVA/

Supervisory Patent Examiner, Art Unit 2449

Application/Control Number: 14/181,112
Art Unit: 2449

Page 19



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
 Address: COMMISSIONER FOR PATENTS
 P.O. Box 1450
 Alexandria, Virginia 22313-1450
 www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
14/181,112	02/14/2014	Zvi Tannenbaum	ZTANN.002P1C3	6874
20995 7590 11/16/2016 KNOBBE MARTENS OLSON & BEAR LLP 2040 MAIN STREET FOURTEENTH FLOOR IRVINE, CA 92614			EXAMINER ASRES, HERMON	
			ART UNIT	PAPER NUMBER
			2449	
			NOTIFICATION DATE	DELIVERY MODE
			11/16/2016	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

jayna.cartee@knobbe.com
 efiling@knobbe.com

<i>Applicant-Initiated Interview Summary</i>	Application No. 14/181,112	Applicant(s) TANNENBAUM ET AL.	
	Examiner HERMON ASRES	Art Unit 2449	

All participants (applicant, applicant's representative, PTO personnel):

(1) HERMON ASRES. (3) Lance Smemoe (REG NO 66,152).

(2) _____. (4) _____.

Date of Interview: 07 November 2016.

Type: ☒ Telephonic ☐ Video Conference
☐ Personal [copy given to: ☐ applicant ☐ applicant's representative]

Exhibit shown or demonstration conducted: ☐ Yes ☒ No.
If Yes, brief description: _____.

Issues Discussed ☒ 101 ☒ 112 ☒ 102 ☐ 103 ☐ Others
(For each of the checked box(es) above, please describe below the issue and detailed description of the discussion)

Claim(s) discussed: 2.

Identification of prior art discussed: Block.

Substance of Interview
(For each issue discussed, provide a detailed description and indicate if agreement was reached. Some topics may include: identification or clarification of a reference or a portion thereof, claim interpretation, proposed amendments, arguments of any applied references etc...)

Applicant discussed proposed amendment to independent claim 1 including adding "a mechanism for the nodes to communicate with each other using a peer-to-peer architecture". Applicant also added a non-transitory language to the independent claim to address the 101 rejection. Applicant further indicated that the "module configured to" will be rewritten to overcome the 112 rejection. Examiner reminded applicant about the pending double patenting rejection and applicant indicated that if the proposed amendment when filed doesn't overcome the double patenting rejection a terminal disclaimer will be filed. No agreement was reached.

Applicant recordation instructions: The formal written reply to the last Office action must include the substance of the interview. (See MPEP section 713.04). If a reply to the last Office action has already been filed, applicant is given a non-extendable period of the longer of one month or thirty days from this interview date, or the mailing date of this interview summary form, whichever is later, to file a statement of the substance of the interview

Examiner recordation instructions: Examiners must summarize the substance of any interview of record. A complete and proper recordation of the substance of an interview should include the items listed in MPEP 713.04 for complete and proper recordation including the identification of the general thrust of each argument or issue discussed, a general indication of any other pertinent matters discussed regarding patentability and the general results or outcome of the interview, to include an indication as to whether or not agreement was reached on the issues raised.

☒ Attachment

/HERMON ASRES/ Examiner, Art Unit 2449	/VIVEK SRIVASTAVA/ Supervisory Patent Examiner, Art Unit 2449
---	--

Manual of Patent Examining Procedure (MPEP), Section 713.04, Substance of Interview Must be Made of Record

A complete written statement as to the substance of any face-to-face, video conference, or telephone interview with regard to an application must be made of record in the application whether or not an agreement with the examiner was reached at the interview.

Title 37 Code of Federal Regulations (CFR) § 1.133 Interviews
Paragraph (b)

In every instance where reconsideration is requested in view of an interview with an examiner, a complete written statement of the reasons presented at the interview as warranting favorable action must be filed by the applicant. An interview does not remove the necessity for reply to Office action as specified in §§ 1.111, 1.135. (35 U.S.C. 132)

37 CFR §1.2 Business to be transacted in writing.

All business with the Patent or Trademark Office should be transacted in writing. The personal attendance of applicants or their attorneys or agents at the Patent and Trademark Office is unnecessary. The action of the Patent and Trademark Office will be based exclusively on the written record in the Office. No attention will be paid to any alleged oral promise, stipulation, or understanding in relation to which there is disagreement or doubt.

The action of the Patent and Trademark Office cannot be based exclusively on the written record in the Office if that record is itself incomplete through the failure to record the substance of interviews.

It is the responsibility of the applicant or the attorney or agent to make the substance of an interview of record in the application file, unless the examiner indicates he or she will do so. It is the examiner's responsibility to see that such a record is made and to correct material inaccuracies which bear directly on the question of patentability.

Examiners must complete an Interview Summary Form for each interview held where a matter of substance has been discussed during the interview by checking the appropriate boxes and filling in the blanks. Discussions regarding only procedural matters, directed solely to restriction requirements for which interview recordation is otherwise provided for in Section 812.01 of the Manual of Patent Examining Procedure, or pointing out typographical errors or unreadable script in Office actions or the like, are excluded from the interview recordation procedures below. Where the substance of an interview is completely recorded in an Examiners Amendment, no separate Interview Summary Record is required.

The Interview Summary Form shall be given an appropriate Paper No., placed in the right hand portion of the file, and listed on the "Contents" section of the file wrapper. In a personal interview, a duplicate of the Form is given to the applicant (or attorney or agent) at the conclusion of the interview. In the case of a telephone or video-conference interview, the copy is mailed to the applicant's correspondence address either with or prior to the next official communication. If additional correspondence from the examiner is not likely before an allowance or if other circumstances dictate, the Form should be mailed promptly after the interview rather than with the next official communication.

The Form provides for recordation of the following information:

- Application Number (Series Code and Serial Number)
- Name of applicant
- Name of examiner
- Date of interview
- Type of interview (telephonic, video-conference, or personal)
- Name of participant(s) (applicant, attorney or agent, examiner, other PTO personnel, etc.)
- An indication whether or not an exhibit was shown or a demonstration conducted
- An identification of the specific prior art discussed
- An indication whether an agreement was reached and if so, a description of the general nature of the agreement (may be by attachment of a copy of amendments or claims agreed as being allowable). Note: Agreement as to allowability is tentative and does not restrict further action by the examiner to the contrary.
- The signature of the examiner who conducted the interview (if Form is not an attachment to a signed Office action)

It is desirable that the examiner orally remind the applicant of his or her obligation to record the substance of the interview of each case. It should be noted, however, that the Interview Summary Form will not normally be considered a complete and proper recordation of the interview unless it includes, or is supplemented by the applicant or the examiner to include, all of the applicable items required below concerning the substance of the interview.

A complete and proper recordation of the substance of any interview should include at least the following applicable items:

- 1) A brief description of the nature of any exhibit shown or any demonstration conducted,
- 2) an identification of the claims discussed,
- 3) an identification of the specific prior art discussed,
- 4) an identification of the principal proposed amendments of a substantive nature discussed, unless these are already described on the Interview Summary Form completed by the Examiner,
- 5) a brief identification of the general thrust of the principal arguments presented to the examiner,
(The identification of arguments need not be lengthy or elaborate. A verbatim or highly detailed description of the arguments is not required. The identification of the arguments is sufficient if the general nature or thrust of the principal arguments made to the examiner can be understood in the context of the application file. Of course, the applicant may desire to emphasize and fully describe those arguments which he or she feels were or might be persuasive to the examiner.)
- 6) a general indication of any other pertinent matters discussed, and
- 7) if appropriate, the general results or outcome of the interview unless already described in the Interview Summary Form completed by the examiner.

Examiners are expected to carefully review the applicant's record of the substance of an interview. If the record is not complete and accurate, the examiner will give the applicant an extendable one month time period to correct the record.

Examiner to Check for Accuracy

If the claims are allowable for other reasons of record, the examiner should send a letter setting forth the examiner's version of the statement attributed to him or her. If the record is complete and accurate, the examiner should place the indication, "Interview Record OK" on the paper recording the substance of the interview along with the date and the examiner's initials.

M E M O R A N D U M

TO: Examiner Hermon Asres
direct: 571-272-4257
email: hermon.asres@uspto.gov

FROM: Lance Smemoe, Reg. #66,152

RE: Agenda for telephone interview on November 7, 2016 at 2:00 p.m. Eastern time

Application No.: 14/181,112

Attorney Docket No.: ZTANN.002P1C3

INFORMAL DOCUMENT FOR DISCUSSION ONLY – DO NOT ENTER

A. Rejections under 35 USC § 112 of claims 2, 6, and 7

Proposed claim amendments

2. (Currently amended) A computer cluster comprising:

a plurality of nodes, wherein one or more of the nodes receives a command to start a cluster initialization process for the computer cluster, and wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel ~~module~~ configured to interpret user instructions; and

a mechanism for the nodes to communicate with each other using a peer-to-peer architecture;

wherein at least one of the nodes returns a result to a user interface module or a script.

6. (Currently amended) The computer cluster of claim 2, wherein each single-node kernel ~~module~~ is stored in a non-transitory computer-readable medium and configured to accept and execute a request.

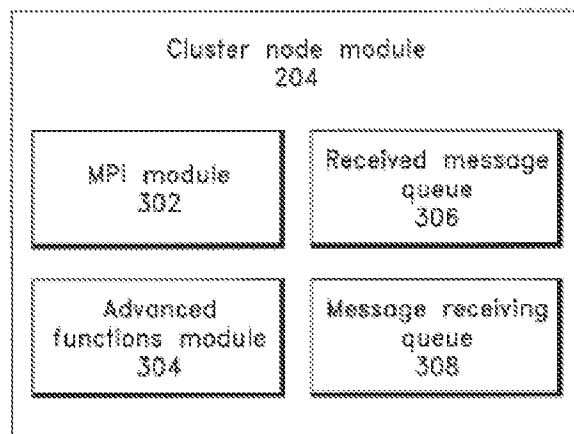
7. (Currently amended) The computer cluster of claim 6, wherein each of the nodes comprises one or more cluster node modules, and each of the cluster node modules is stored in a non-transitory computer-readable medium and configured to communicate with the single-node kernel and with one or more other cluster node modules.

M E M O R A N D U M

Page 2

INFORMAL DOCUMENT FOR DISCUSSION ONLY – DO NOT ENTER*Non-limiting example embodiments from the specification*

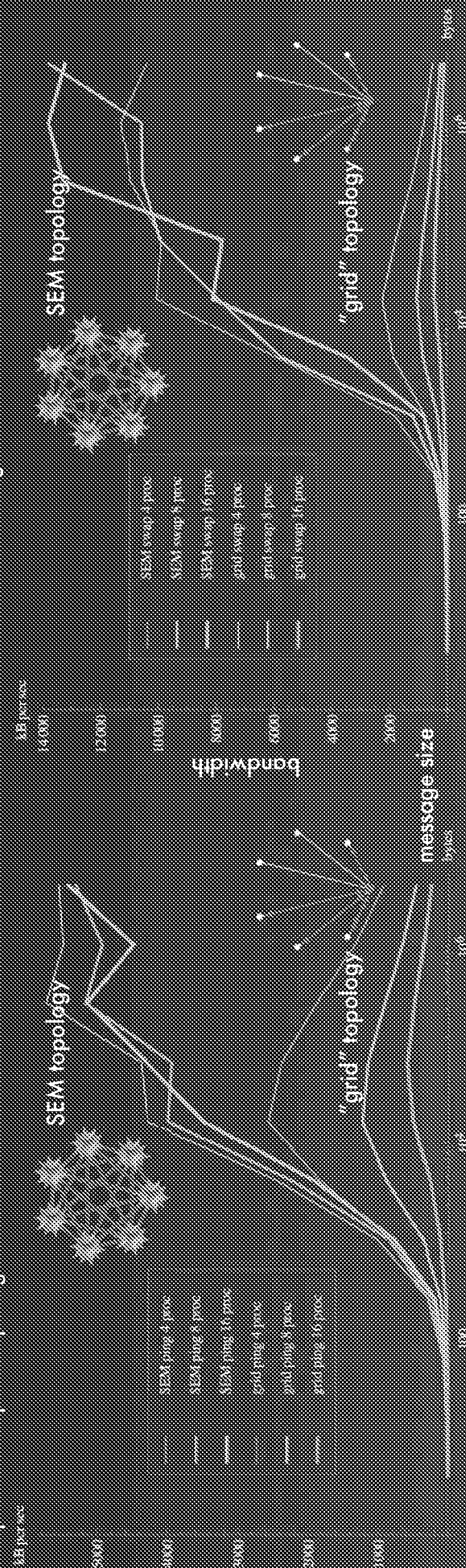
[0093] A kernel module 206 typically includes program code for interpreting high-level code, commands, and/or instructions supplied by a user or a script into low-level code, such as, for example, machine language or assembly language. In one embodiment, each cluster node module 204a-e is connected to all other cluster node modules, while each kernel module 206a-e is allocated and connected only to one cluster node module 204. In one embodiment, there is one cluster node module-kernel module pair per processor. For example, in an embodiment of a computer cluster 100 including single-processor computer systems, each cluster node module-kernel module pair could reside on a single-processor computer. If a computer contains multiple processors or processing cores, it may contain multiple cluster node module-kernel module pairs, but the pairs can still communicate over the cluster node module's network connections.

*FIG. 3***B. Rejections under 35 USC § 101 of Claims 2-11**

Pingpong MPI Benchmark - SEM vs. "grid"

The Supercomputing Engine for Mathematica (SEM), like modern supercomputers, supports an "all-to-all" communications network between *Mathematica* kernels. Practitioners in high-performance computing know the communications network ultimately limits the addressable problem size. SEM provides supercomputing infrastructure for Wolfram's *Mathematica*.

We tested the performance of SEM versus grid/*Mathematica* using the pingpong and swap communications benchmarks on a cluster with 4, 8, and 16 processors. While SEM's performance remained steady for increased cluster size, grid/*Mathematica*'s performance decreased due to its "master" kernel becoming a bottleneck for all communications.



A high-performance computing benchmark called "pingpong", developed by Viktor K. Decyk of the UCLA Plasma Physics Group, stresses the network of a parallel computer in two stages after dividing the processing elements into the even and odd processors. The pingpong stage has each even one send a message to an odd one (0 to 1, 2 to 3, ...) and back again. The message is varied in size from a few bytes to many megabytes while each operation is timed and averaged. The second stage performs a "swap", meaning each pair of processes sends and receives messages from each other simultaneously, if possible. Again the message size is varied and the results are timed and averaged.

ZTANN.002P1C3

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

First Inventor	: Zvi Tannenbaum
App. No.	: 14/181,112
Filed	: February 14, 2014
For	: CLUSTER COMPUTING
Examiner	: Asres, Hermon
Art Unit	: 2449
Conf. No.	: 6874

RESPONSE TO OFFICE ACTION DATED JULY 20, 2016

Mail Stop Amendment

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

In response to the Office Action dated July 20, 2016, please reconsider the rejections in view of the following remarks.

Amendments to the Claims are reflected in the listing of claims which begins on page 2.

Summary of Interview begins on page 7.

Remarks/Arguments begin on page 8.

Application No.: 14/181,112
Filing Date: February 14, 2014

AMENDMENTS TO THE CLAIMS

1. (Canceled)
2. (Currently amended) A computer cluster comprising:
a plurality of nodes comprising a hardware processor, wherein one or more of the nodes receives a command to start a cluster initialization process for the computer cluster, and wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, causes the hardware processor ~~module configured~~ to interpret user instructions; and
a mechanism for the nodes to communicate with each other using a peer-to-peer architecture;
wherein at least one of the nodes returns a result to a user interface or a script.
3. (Currently amended) The computer cluster of claim 2, wherein the single-node kernel ~~module~~ comprises a Mathematica kernel.
4. (Previously presented) The computer cluster of claim 2, wherein the mechanism comprises a message passing interface.
5. (Previously presented) The computer cluster of claim 2, wherein each of the nodes comprises one or more cluster node modules.
6. (Currently amended) The computer cluster of claim 2, wherein each single-node kernel ~~module~~ is stored in a non-transitory computer-readable medium and configured to accept and execute a request.
7. (Currently amended) The computer cluster of claim 6, wherein each of the nodes comprises one or more cluster node modules, ~~[[and]]~~ wherein each of the cluster node modules ~~[[is]]~~ comprises instructions stored in a non-transitory computer-readable medium, and configured to ~~wherein the instructions, when executed by the hardware processor, cause the cluster node module to~~ communicate with the single-node kernel and with one or more other cluster node modules.
8. (Previously presented) The computer cluster of claim 7, wherein the plurality of cluster node modules act as a cluster.
9. (Previously presented) The computer cluster of claim 7, wherein the plurality of cluster node modules communicate with one another to act as a cluster.

Application No.: 14/181,112
Filing Date: February 14, 2014

10. (Currently amended) The computer cluster of claim 9, wherein the computer cluster includes[[a]]the user interface.

11. (Previously presented) The computer cluster of claim 10, wherein each cluster node module accepts instructions from the user interface and interprets one or more of the instructions.

12. (New) The computer cluster of claim 2, wherein the cluster initialization process comprises establishing communication among two or more of the nodes.

13. (New) The computer cluster of claim 2, wherein the cluster initialization process comprises configuring access by one or more of the nodes to a computer-readable medium comprising program code for the single-node kernel.

14. (New) The computer cluster of claim 13, wherein the cluster initialization process comprises establishing message-passing support among the nodes in the cluster.

15. (New) The computer cluster of claim 13, wherein the cluster initialization process comprises launching cluster node modules by the cluster.

16. (New) The computer cluster of claim 15, wherein the cluster initialization process comprises assigning a processor identification number to each of the cluster node modules.

17. (New) The computer cluster of claim 2, wherein the cluster initialization process comprises:

launching cluster node modules by the cluster; and

after launching the cluster node modules, configuring access by one or more of the nodes to a non-transitory computer-readable medium comprising program code for the single-node kernel.

18. (New) The computer cluster of claim 2, wherein the cluster initialization process comprises:

launching cluster node modules by the cluster;

after launching the cluster node modules, establishing communication among two or more of the nodes; and

after establishing communication among two or more of the nodes, assigning a processor identification number to each of the cluster node modules.

19. (New) The computer cluster of claim 2, wherein one or more of the nodes are configured to communicate at least some of the user instructions.

Application No.: 14/181,112
Filing Date: February 14, 2014

20. (New) The computer cluster of claim 19, wherein one or more of the nodes are configured to communicate at least some of the user instructions to one or more single-node kernels.

21. (New) The computer cluster of claim 2, wherein one or more of the nodes are configured to accept user instructions via one or more of the nodes.

22. (New) The computer cluster of claim 21, wherein one or more of the nodes are configured to communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other.

23. (New) The computer cluster of claim 2, wherein one or more of the nodes are configured to transmit at least some of the user instructions that originate from a user interface.

24. (New) The computer cluster of claim 2, wherein one or more of the nodes are configured to parallelize at least some of the user instructions before communicating at least some of the user instructions to one or more single-node kernels.

25. (New) The computer cluster of claim 2, wherein one or more of the nodes are configured to accept user instructions before communicating at least some of the user instructions to one or more single-node kernels.

26. (New) The computer cluster of claim 2, wherein one or more of the nodes are configured to:

accept user instructions;

after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and

after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to one or more single-node kernels.

27. (New) The computer cluster of claim 2, wherein the plurality of nodes are configured to communicate with one another to interpret and translate commands for execution by a plurality of single-node kernels.

28. (New) A computer cluster comprising:

a plurality of nodes, wherein one or more of the nodes are configured to receive:

a command to start a cluster initialization process for the computer cluster, wherein the cluster initialization process comprises establishing communication among two or more of the nodes; and

Application No.: 14/181,112
Filing Date: February 14, 2014

an instruction from a user interface or a script;
and
a mechanism for the nodes to communicate with each other using asynchronous calls;

wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel configured to interpret user instructions; and

wherein at least one of the nodes returns a result to the user interface or the script.

29. (New) The computer cluster of claim 28, wherein the asynchronous calls comprise a first command to create a first packet containing:

an expression to be sent as payload; and

a target node where the expression should be sent;

wherein the first command is configured to be called from within a single-node kernel;

wherein the single-node kernel is configured to send the first packet to a local cluster node module; and

wherein the local cluster node module is configured to forward the expression to the target node.

30. (New) The computer cluster of claim 29, wherein the asynchronous calls comprise a second command to create a second packet containing:

a location where the expression is expected to be received; and

a sending node from which the expression is expected to be received;

wherein the second command is configured to be called from within a single-node kernel;

wherein the single-node kernel is configured to send the second packet to a local cluster node module; and

wherein the local cluster node module is configured to store the second packet contents in a message receiving queue.

31. (New) A computer cluster comprising:

a plurality of nodes, wherein one or more of the nodes are configured to receive:

Application No.: 14/181,112
Filing Date: February 14, 2014

a command to start a cluster initialization process for the computer cluster, wherein the cluster initialization process comprises establishing communication among two or more of the nodes; and

an instruction from a user interface or a script;

and

a mechanism for the nodes to communicate with each other;

wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel configured to interpret user instructions and interpret the user instructions into commands that are executable by a special purpose microprocessor; and

wherein at least one of the nodes returns a result to the user interface or the script.

Application No.: 14/181,112
Filing Date: February 14, 2014

SUMMARY OF INTERVIEW

Attendees, Date and Type of Interview

A telephonic interview was conducted on November 7, 2016 and attended by Examiner Hermon Asres and Applicant's representative, Lance D. Smemoe (reg. no. 66,152).

Exhibits and/or Demonstrations

None

Identification of Claims Discussed

Claims 2-11

Identification of Cited/Disclosed Art Discussed

None

Proposed Amendments

Substantially as set forth herein

Principal Arguments and Other Matters

Applicant's representative presented proposed amendments and argued that the rejections under 35 USC § 112 and 35 USC § 101 are inapplicable to the claims as amended. Specific arguments are set forth in the following remarks.

Results of Interview

Applicant agreed to submit a formal response with claim amendments. The Examiner will reconsider the rejections and confirm the allowability of the amended claims.

Application No.: 14/181,112
Filing Date: February 14, 2014

REMARKS

In the Office Action dated July 20, 2016, Claims 2-11 were rejected, as discussed below. In this Amendment, claims 2, 3, 6, 7, and 10 are amended, and new claims 12-31 are added. No new matter is added. After the amendments, claims 2-31 are pending for consideration. Claims 2, 28, and 31 are independent claims.

Telephone Interview

Applicant thanks the Examiner for the courteous and helpful telephone interview held on November 7, 2016 (summarized above).

Response to Double Patenting Rejection

Claim 2 is rejected on the ground of non-statutory (obviousness-type) double patenting over claims 1 and 8 of U.S. Patent No. 8,676,877. Applicant respectfully requests that the double patenting rejection be held in abeyance until the other rejections are overcome.

Response to Rejections under 35 U.S.C. § 112

Claims 2, 6, and 7 are rejected under 35 U.S.C. § 112 as allegedly being indefinite for failing to particularly point out and distinctly claim the subject matter. Applicant respectfully disagrees with the rejections because the specification discloses structure, material, or acts that correspond to the claimed subject matter. Nevertheless, to expedite allowance of the application, Applicant has amended claims 2, 6, and 7 to remove the language identified by the examiner as invoking 35 U.S.C. § 112, sixth paragraph. Accordingly, Applicant respectfully requests that the rejections be withdrawn.

Response to Rejections under 35 U.S.C. § 101

Claims 2-11 are rejected under 35 U.S.C. § 101 as allegedly being directed to non-statutory subject matter. Applicant respectfully traverses the rejection. The Supreme Court of the United States has stated that a claim that recites “an improvement to the functioning of the computer itself” is directed to significantly more than an abstract idea. *Alice Corp. v. CLS Bank*, 134 S. Ct. at 2359. Claim 2 as amended recites improvements that improve operation of the claimed computer cluster. For example, claim 2 recites that “the nodes . . . communicate with

Application No.: 14/181,112
Filing Date: February 14, 2014

each other using a peer-to-peer architecture.” The peer-to-peer architecture improves operation of the computer because it makes intermediate results of computation available to other nodes, thus permitting some computations to be solved using multi-node processing that are not possible with other types of distributed computing. *See, e.g.*, paragraphs [0005] and [0062] of the application as filed. As another example, the peer-to-peer architecture can increase performance of the computer compared to other architectures. *See, e.g.*, “Pingpong MPI Benchmark – SEM vs ‘grid’,” cited as reference no. 3 in the Information Disclosure Statement filed herewith. This information demonstrates that claim 2 recites an improvement to the functioning of the computer itself and, thus, meets the requirements of subject-matter patent-eligibility under 35 U.S.C. § 101. Applicant respectfully requests withdrawal of the rejections.

Response to Rejections under 35 U.S.C. § 102 and 35 U.S.C. § 103

The Office Action rejects Claims 2, and 4-11 under 35 U.S.C. § 102, arguing that they are anticipated by U.S. Patent Publication No. 2005/0021751 to Block et al. (hereafter “Block”). Applicant respectfully traverses the rejections.

The Office Action rejects Claim 3 under 35 U.S.C. § 103(a), arguing that it is unpatentable in view of Block and U.S. Patent Publication No. 2007/0073705 to Gray. Applicant respectfully traverses the rejection.

Applicant respectfully submits that Block does not disclose every feature recited in claim 2 or the claims that depend from claim 2. For example, Block does not disclose at least a computer cluster comprising “a mechanism for the nodes to communicate with each other using a peer-to-peer architecture,” in combination with the other features recited in claim 2. As another example, Block does not disclose at least that “at least one of the nodes returns a result to a user interface or a script,” in combination with the other features recited in claim 2. Indeed, Block does not disclose that the disclosed target nodes and backup nodes share data with one another using a peer-to-peer architecture.

Each of dependent Claims 3-11 is patentable because each depends directly or indirectly from claim 2 and because of the additional features recited in each claim.

For at least these reasons, Applicant respectfully submits that Claims 2-11 are allowable over the cited references and respectfully requests that the rejections of these claims be withdrawn.

Application No.: 14/181,112
Filing Date: February 14, 2014

New claims 12-31

New claims 12-31 are added in this amendment. No new matter is added. Each of claims 12-27 depends directly or indirectly from claim 2 and is patentable for that reason and for the additional features recited in each claim. Each of claims 28-31 is patentable because the cited references do not anticipate or render obvious the unique combination of features recited in each of claims 28-31.

No Disclaimers or Disavowals

Applicant respectfully submits that the claims are in condition for allowance. Furthermore, any remarks in support of patentability of one claim should not be imputed to any other claim, even if similar terminology is used. Any remarks referring to only a portion of a claim should not be understood to base patentability on that portion or that the limitation discussed is essential or critical; rather, patentability must rest on each claim taken as a whole. Applicant respectfully traverses each of the Examiner's rejections and each of the Examiner's assertions regarding what the prior art shows or teaches, even if not expressly discussed herein. Although the present communication may include alterations to the application or claims, or characterizations of claim scope or referenced art, applicant is not conceding in this application that previously pending claims are not patentable over the cited references. Rather, any alterations or characterizations are being made to facilitate expeditious prosecution of this application. Applicant reserves the right to pursue at a later date any previously pending or other broader or narrower claims that capture any subject matter supported by the present disclosure, including subject matter found to be specifically disclaimed herein or by any prior prosecution. Accordingly, reviewers of this or any parent, child or related prosecution history shall not reasonably infer that applicant has made any disclaimers or disavowals of any subject matter supported by the present application

CONCLUSION

For the foregoing reasons, it is respectfully submitted that the rejections set forth in the outstanding Office Action are inapplicable to the present claims. Accordingly, issuance of a Notice of Allowance is requested.

Application No.: 14/181,112
Filing Date: February 14, 2014

The undersigned has made a good faith effort to respond to all of the rejections in the case and to place the claims in condition for immediate allowance. Nevertheless, if any undeveloped issues remain or if any issues require clarification, the Examiner is respectfully requested to call the undersigned below.

Please charge any additional fees, including any fees for additional extension of time, or credit overpayment to Deposit Account No. 11-1410.

Respectfully submitted,

KNOBBE, MARTENS, OLSON & BEAR, LLP

Dated: November 21, 2016

By: /Lance D. Smemoe/

Lance D. Smemoe
Registration No. 66,152
Attorney of Record
Customer No. 20995
(949) 760-0404

24428740



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
 Address: COMMISSIONER FOR PATENTS
 P.O. Box 1450
 Alexandria, Virginia 22313-1450
 www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
14/181,112	02/14/2014	Zvi Tannenbaum	ZTANN.002P1C3	6874
20995 7590 03/09/2017 KNOBBE MARTENS OLSON & BEAR LLP 2040 MAIN STREET FOURTEENTH FLOOR IRVINE, CA 92614			EXAMINER ASRES, HERMON	
			ART UNIT	PAPER NUMBER
			2449	
			NOTIFICATION DATE	DELIVERY MODE
			03/09/2017	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

jayna.cartee@knobbe.com
 efiling@knobbe.com

Office Action Summary**Application No.**
14/181,112**Applicant(s)**
TANNENBAUM ET AL.**Examiner**
HERMON ASRES**Art Unit**
2449**AIA (First Inventor to File)
Status**
No**-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --****Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTHS FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 11/21/2016.
☐ A declaration(s)/affidavit(s) under **37 CFR 1.130(b)** was/were filed on ____.
- 2a) ☒ This action is **FINAL**. 2b) ☐ This action is non-final.
- 3) ☐ An election was made by the applicant in response to a restriction requirement set forth during the interview on ____; the restriction requirement and election have been incorporated into this action.
- 4) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims*

- 5) ☒ Claim(s) 2-31 is/are pending in the application.
 5a) Of the above claim(s) ____ is/are withdrawn from consideration.
- 6) ☐ Claim(s) ____ is/are allowed.
- 7) ☒ Claim(s) 2-31 is/are rejected.
- 8) ☐ Claim(s) ____ is/are objected to.
- 9) ☐ Claim(s) ____ are subject to restriction and/or election requirement.

* If any claims have been determined allowable, you may be eligible to benefit from the **Patent Prosecution Highway** program at a participating intellectual property office for the corresponding application. For more information, please see http://www.uspto.gov/patents/init_events/pph/index.jsp or send an inquiry to PPHfeedback@uspto.gov.

Application Papers

- 10) ☐ The specification is objected to by the Examiner.
- 11) ☐ The drawing(s) filed on ____ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.
 Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
 Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

Certified copies:

- a) ☐ All b) ☐ Some** c) ☐ None of the:
1. ☐ Certified copies of the priority documents have been received.
 2. ☐ Certified copies of the priority documents have been received in Application No. ____.
 3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

** See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- 1) ☒ Notice of References Cited (PTO-892)
- 2) ☒ Information Disclosure Statement(s) (PTO/SB/08a and/or PTO/SB/08b)
 Paper No(s)/Mail Date ____.
- 3) ☐ Interview Summary (PTO-413)
 Paper No(s)/Mail Date. ____.
- 4) ☐ Other: ____.

Application/Control Number: 14/181,112
Art Unit: 2449

Page 2

DETAILED ACTION

The following is a Final office action in response to the Amendments filed on November 21, 2016.

Claims 2, 3, 6, 7, and 10 have been amended.

Claims 12-31 have been added.

Claims 2-31 are pending.

Response to Arguments

Double Patenting Rejection

The double patenting rejection will not be withdrawn because applicant in remarks indicated that the double patenting rejection be held in abeyance until the other rejections are overcome.

35 U.S.C. 112 Rejections

Examiner withdraws the 112 rejection of claims 2, 6, and 7 in view of applicant's amendment/remarks filed on 11/21/2016.

Application/Control Number: 14/181,112
Art Unit: 2449

Page 3

35 U.S.C. 101 Rejections

Examiner withdraws the 101 rejection of claims 2-11 (previously rejected for being directed to a judicial exception without significantly more) in view of applicant's amendment/remarks filed on 11/21/2016.

35 U.S.C. 103 Rejections

Applicant's argument filed 11/21/2016 has been fully considered but they are deemed not persuasive in view of new grounds of rejection.

Double Patenting

The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. A nonstatutory double patenting rejection is appropriate where the claims at issue are not identical, but at least one examined application claim is not patentably distinct from the reference claim(s) because the examined application claim is either anticipated by, or would have been obvious over, the reference claim(s). See, e.g., *In re Berg*, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*,

Application/Control Number: 14/181,112

Page 4

Art Unit: 2449

686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the reference application or patent either is shown to be commonly owned with this application, or claims an invention made as a result of activities undertaken within the scope of a joint research agreement. A terminal disclaimer must be signed in compliance with 37 CFR 1.321(b).

The USPTO internet Web site contains terminal disclaimer forms which may be used. Please visit <http://www.uspto.gov/forms/>. The filing date of the application will determine what form should be used. A web-based eTerminal Disclaimer may be filled out completely online using web-screens. An eTerminal Disclaimer that meets all requirements is auto-processed and approved immediately upon submission. For more information about eTerminal Disclaimers, refer to <http://www.uspto.gov/patents/process/file/efs/guidance/eTD-info-I.jsp>.

Claim 2 is rejected on the ground of nonstatutory double patenting as being unpatentable over claims 1 and 8 of U.S. Patent No. 8,676,877. Although the conflicting claims are not identical, they are not patentably distinct from each other because:

Application/Control Number: 14/181,112

Page 5

Art Unit: 2449

Instant Application		US Patent 8,676,877	
Claim	Limitation	Claim	Limitation
2, 28, and 31	<p>A computer cluster comprising:</p> <p>a plurality of nodes, comprising a hardware processor wherein one or more of the nodes receives a command to start a cluster initialization process for the computer cluster, and</p> <p>wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that when executed causes the hardware processor to interpret user instructions; and</p> <p>a mechanism for the nodes to communicate with each other using a peer-to-peer architecture; wherein at least one of the nodes returns a result to a user interface or a script</p>	1 and 8	<p>A system for performing an instruction received from a front end by executing commands on one or more special purpose microprocessors, the system comprising:</p> <p>a plurality of nodes, wherein each node is configured to access a computer-readable memory system comprising program code for a single-node kernel module,</p> <p>wherein each cluster node module is stored in a computer-readable memory system and configured to communicate with a single-node kernel and with one or more other cluster node modules, to accept instructions, and to interpret at least some of the instructions such that the plurality of cluster node modules communicate with one another in order to act as a cluster in executing commands using one or more hardware processors</p>

Application/Control Number: 14/181,112
Art Unit: 2449

Page 6

Claim Rejections - 35 USC § 103

The following is a quotation of pre-AIA 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

Claims 2, and 4-31 are rejected under 35 U.S.C. 103(a) as being unpatentable over Block et al. (U.S. PGPub 2005/0021751), in view of Singh (US Patent 8,601,101).

As per claim 2, Block teaches a computer cluster (Block,cluster data port services within a cluster infrastructure. See paragraph [0025]) comprising:

a plurality of nodes, comprising a hardware processor wherein one or more of the nodes receives a command to start a cluster initialization process for the computer cluster, and wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel (Block,multiple logical or TCP connections to a designated target node with designated backup target nodes, which appears to the user (i.e., an operating system or ***kernel component***) ***like a single data port or data pipe***. See paragraph [0044]) that when executed causes the hardware processor to interpret user instructions; and a mechanism for the nodes to communicate with each other. (Block, establishment of multiple network connections between a source node, a target node and one or more

Application/Control Number: 14/181,112

Page 7

Art Unit: 2449

backup nodes in such a manner that a cluster data port is effectively utilized as single data port. See paragraph [0025]).

Block doesn't explicitly teach that the nodes communicate with each other using a peer-to-peer architecture and also that one of the nodes returns a result to a user interface or a script. In analogous art Singh teaches nodes that communicate with each other using a peer-to-peer architecture. (Singh,each node in a cluster including a peer-to-peer communication channel compiling and maintaining its own cluster membership are disclosed..... each cluster node may be coupled to every other node and commonly-accessible storage through a network.Based on the cluster membership information, the joining node may request a peer-to-peer connection with each potential cluster member. See Column 2 line 26-38). Singh also teaches one of the nodes returns a result to a user interface or a script (Singh, The cluster configuration service 1510A may include a user interface to allow a system administrator to perform various node management functions through administrative console 1500. When the administrator makes changes to the node 240A configuration, the cluster configuration service 1510A may record all of the modifications that are made. See Column 14 line 61-66).

Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to take the teaching of Singh and apply them on the teaching of Block because when each node on the cluster communication channel is in peer-to-peer connection, each node can detect the connection failure based on the socket communication with the failed node and can update its cluster membership data without

Application/Control Number: 14/181,112

Page 8

Art Unit: 2449

the need for a central entity to pass an updated membership list among nodes. (Singh, See column 13 line 23-28).

As per claim 4, Block-Singh teaches wherein the mechanism comprises a message passing interface. (Block, a single data port instance may also support target to source message sends. See paragraph [0050]).

As per claim 5, Block-Singh teaches wherein each of the nodes comprises one or more cluster node modules. (Block, cluster data port services within a cluster infrastructure to provide reliable and efficient communications between nodes. See paragraph [0010]).

As per claim 6, Block-Singh teaches wherein each single-node kernel is stored in a non-transitory computer-readable medium and configured to accept and execute a request. (Block, The routines executed to implementoperating system or a specific application, component, program, object, module or sequence of instructions, will also be referred to herein as "computer program code," or simply "program code." See paragraph [0039], [0049]).

As per claim 7, Block-Singh teaches wherein each of the nodes comprises one or more cluster node modules, wherein each of the cluster node modules comprises instructions stored in a non-transitory computer- readable medium and wherein the

Application/Control Number: 14/181,112

Page 9

Art Unit: 2449

instructions when executed by the hardware processor cause the cluster node module to communicate with the single-node kernel and with one or more other cluster node modules. (Block,multiple logical or TCP connections to a designated target node with designated backup target nodes, which appears to the user (i.e., an operating system or **kernel component**) **like a single data port or data pipe**. See paragraph [0044], [0039]).

As per claim 8, Block-Singh teaches wherein the plurality of cluster node modules act as a cluster. (Block, utilize cluster data port services within a cluster infrastructure to provide reliable and efficient communications between nodes in a clustered computer system. See paragraph [0010]).

As per claim 9, Block-Singh teaches wherein the plurality of cluster node modules communicate with one another to act as a cluster. (Block, Nodes 12, 14 and 16 are typically coupled together via a clustering network 18. See paragraph [0028]).

As per claim 10, Block-Singh teaches wherein the computer cluster includes the user interface. (Block, Cluster manager application 60 that provides the user interface whereby a user such as a systems administrator can manage clustering operations in the system. See paragraph [0037]).

Application/Control Number: 14/181,112
Art Unit: 2449

Page 10

As per claim 11, Block-Singh teaches wherein each cluster node module accepts instructions from the user interface and interprets one or more of the instructions. (Block, Cluster manager application 60 that provides the user interface whereby a user such as a systems administrator can manage clustering operations in the system. See paragraph [0037], [0039]).

As per claim 12, Block-Singh teaches wherein the cluster initialization process comprises establishing communication among two or more of the nodes. (Block, communications between nodes in a clustered computer system. See paragraph [0010]).

As per claim 13, Block-Singh teaches wherein the cluster initialization process comprises configuring access by one or more of the nodes to a computer-readable medium comprising program code for the single-node kernel. (Block, The cluster data port services described hereinafter, on the other hand, provide an abstracted transport service that encapsulates and manages the establishment of multiple logical or TCP connections to a designated target node with designated backup target nodes, which appears to the user (i.e., an operating system or kernel component) like a single data port or data pipe. See paragraph [0044]).

As per claim 14, Block-Singh teaches wherein the cluster initialization process comprises establishing message-passing support among the nodes in the cluster.

Application/Control Number: 14/181,112

Page 11

Art Unit: 2449

(Block, data port services may also be configured to provide synchronous and asynchronous caller send models, as well as support message encryption. As such, the herein-described services may be used to provide a general messaging service that allows a variety of operating system or kernel components to make use of the services in a clustered environment. See paragraph [0045]).

As per claim 15, Block-Singh teaches wherein the cluster initialization process comprises launching cluster node modules by the cluster. (Block, see paragraph [0011], [0012]).

As per claim 16, Block-Singh teaches wherein the cluster initialization process comprises assigning a processor identification number to each of the cluster node modules. (Block, Data port internal object 72 also has a data port address table 78, which stores remote IP address, node ID pairs relating to the available IP addresses through which a particular node (identified by the node ID) is coupled to the clustering network. See paragraph [0059]).

As per claim 17, Block-Singh teaches wherein the cluster initialization process comprises: launching cluster node modules by the cluster; and after launching the cluster node modules, configuring access by one or more of the nodes to a non-transitory computer-readable medium comprising program code for the single-node kernel. (Block, data port services may also be configured to provide synchronous and

Application/Control Number: 14/181,112

Page 12

Art Unit: 2449

asynchronous caller send models, as well as support message encryption. As such, the herein-described services may be used to provide a general messaging service that allows a variety of operating system or kernel components to make use of the services in a clustered environment. See paragraph [0045]).

As per claim 18, Block-Singh teaches wherein the cluster initialization process comprises: launching cluster node modules by the cluster; after launching the cluster node modules, establishing communication among two or more of the nodes; (Block, a cluster data port consistent with the invention may be used to facilitate the transfer of large volumes of data between a source node and specified target nodes in a clustering environment. See paragraph [0012]) and after establishing communication among two or more of the nodes, assigning a processor identification number to each of the cluster node modules. (Block, Data port internal object 72 also has a data port address table 78, which stores remote IP address, node ID pairs relating to the available IP addresses through which a particular node (identified by the node ID) is coupled to the clustering network. See paragraph [0059]).

As per claim 19, Block-Singh teaches wherein one or more of the nodes are configured to communicate at least some of the user instructions. (Block, establishment of multiple network connections between a source node, a target node and one or more backup nodes in such a manner that a cluster data port is effectively utilized as single data port. See paragraph [0025], [0044]).

Application/Control Number: 14/181,112
Art Unit: 2449

Page 13

As per claim 20, Block-Singh teaches wherein one or more of the nodes are configured to communicate at least some of the user instructions to one or more single-node kernel. (Block,multiple logical or TCP connections to a designated target node with designated backup target nodes, which appears to the user (i.e., an operating system or ***kernel component***) ***like a single data port or data pipe***. See paragraph [0044]).

As per claim 21, Block-Singh teaches wherein one or more of the nodes are configured to accept user instructions via one or more of the nodes. (Block, see paragraph [0025], [0044]).

As per claim 22, Block-Singh teaches wherein one or more of the nodes are configured to communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other. (Block, see paragraph [0025], [0044]).

As per claim 23, Block-Singh teaches wherein one or more of the nodes are configured to transmit at least some of the user instructions that originate from a user interface. (Singh, The cluster configuration service 1510A may include a user interface to allow a system administrator to perform various node management functions through administrative console 1500. When the administrator makes changes to the node 240A configuration, the cluster configuration service 1510A may record all of the modifications that are made. See Column 14 line 61-66).

Application/Control Number: 14/181,112
Art Unit: 2449

Page 14

As per claim 24, Block-Singh teaches wherein one or more of the nodes are configured to parallelize at least some of the user instructions before communicating at least some of the user instructions to one or more single-node kernels. (Block,multiple logical or TCP connections to a designated target node with designated backup target nodes, which appears to the user (i.e., an operating system or **kernel component**) **like a single data port or data pipe**. See paragraph [0044]).

As per claim 25, Block-Singh teaches wherein one or more of the nodes are configured to accept user instructions before communicating at least some of the user instructions to one or more single-node kernels. (Singh, The cluster configuration service 1510A may include a user interface to allow a system administrator to perform various node management functions through administrative console 1500. When the administrator makes changes to the node 240A configuration, the cluster configuration service 1510A may record all of the modifications that are made. See Column 14 line 61-66).

As per claim 26, Block-Singh teaches wherein one or more of the nodes are configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; (Block, see paragraph [0025], [0044]). and after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to one or more single-node kernels.

Application/Control Number: 14/181,112

Page 15

Art Unit: 2449

(Block,multiple logical or TCP connections to a designated target node with designated backup target nodes, which appears to the user (i.e., an operating system or ***kernel component***) ***like a single data port or data pipe***. See paragraph [0044]). (Also see Singh Column 14 line 61-66).

As per claim 27, Block-Singh teaches wherein the plurality of nodes are configured to communicate with one another to interpret and translate commands for execution by a plurality of single-node kernels. (Block,multiple logical or TCP connections to a designated target node with designated backup target nodes, which appears to the user (i.e., an operating system or ***kernel component***) ***like a single data port or data pipe***. See paragraph [0044]). (Also see Singh Column 14 line 61-66).

As per claim 28, Block teaches a computer cluster comprising: a plurality of nodes, (Block,cluster data port services within a cluster infrastructure. See paragraph [0025]) wherein one or more of the nodes are configured to receive:

a command to start a cluster initialization process for the computer cluster, wherein the cluster initialization process comprises establishing communication among two or more of the nodes; (Block, establishment of multiple network connections between a source node, a target node and one or more backup nodes in such a manner that a cluster data port is effectively utilized as single data port. See paragraph [0025]).

and a mechanism for the nodes to communicate with each other using asynchronous calls; wherein each of the nodes is configured to access a non-transitory

Application/Control Number: 14/181,112

Page 16

Art Unit: 2449

computer- readable medium comprising program code for a single-node kernel configured to interpret user instructions; and wherein at least one of the nodes returns a result to the user interface or the script. (Block,multiple logical or TCP connections to a designated target node with designated backup target nodes, which appears to the user (i.e., an operating system or ***kernel component***) ***like a single data port or data pipe***. See paragraph [0044]).

Block doesn't explicitly teach receiving an instruction from a user interface or a script. In analogous art Singh teaches receiving instruction from a user interface in a computer cluster. (Singh, The cluster configuration service 1510A may include a user interface to allow a system administrator to perform various node management functions through administrative console 1500. When the administrator makes changes to the node 240A configuration, the cluster configuration service 1510A may record all of the modifications that are made. See Column 14 line 61-66).

Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to take the teaching of Singh and apply them on the teaching of Block because it would allow a system administrator to perform various node management functions through administrative console. (Singh, ... allow a system administrator to perform various node management functions through administrative console 1500. When the administrator makes changes to the node 240A configuration, the cluster configuration service 1510A may record all of the modifications that are made. See column 14 line 61-66).

Application/Control Number: 14/181,112
Art Unit: 2449

Page 17

As per claim 29, Block-Singh teaches wherein the asynchronous calls comprise a first command to create a first packet containing: an expression to be sent as payload; and a target node where the expression should be sent; (Block, provide synchronous and asynchronous caller send models, as well as support message encryption. As such, the herein-described services may be used to provide a general messaging service that allows a variety of operating system or kernel components to make use of the services in a clustered environment. See paragraph [0045]) wherein the first command is configured to be called from within a single-node kernel; wherein the single-node kernel is configured to send the first packet to a local cluster node module; and wherein the local cluster node module is configured to forward the expression to the target node. (Block,multiple logical or TCP connections to a designated target node with designated backup target nodes, which appears to the user (i.e., an operating system or ***kernel component***) ***like a single data port or data pipe***. See paragraph [0044]). (Also see Singh Column 14 line 61-66).

As per claim 30,

[Rejection rational for claim 29 is applicable].

As per claim 31,

[Rejection rational for claim 28 is applicable].

Application/Control Number: 14/181,112
Art Unit: 2449

Page 18

Claim 3 is rejected under 35 U.S.C. 103(a) as being unpatentable over Block et al. (US PGPub 2005/0021751) in view of Singh (US Patent 8,601,101) and further in view of Gray (USPGPub 2007/0073705).

As per claim 3, Block-Singh teaches the computer cluster of claim 2 but doesn't teach a Mathematica kernel. In analogous art Gray teaches a Mathematica kernel. (Gray, The front end may include an interactive document referred to as a notebook similar to those often used with MATHEMATICA.RTM. software systems. A notebook may include input to be sent to the kernel 104 and output received from the kernel, as well as text, graphics, palettes, etc. see paragraph [0031]).

Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to take the teaching of Gray and apply them on the teaching of Block because the kernel and the front end may be implemented on a same computing system or on different computing systems that are communicatively coupled to one another. (Gray, see paragraph [0030]).

Conclusion

THIS ACTION IS MADE FINAL. Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not

Application/Control Number: 14/181,112

Page 19

Art Unit: 2449

mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to HERMON ASRES whose telephone number is (571)272-4257. The examiner can normally be reached on Monday - Friday 7:30 am to 5:00 pm est.

Examiner interviews are available via telephone, in-person, and video conferencing using a USPTO supplied web-based collaboration tool. To schedule an interview, applicant is encouraged to use the USPTO Automated Interview Request (AIR) at <http://www.uspto.gov/interviewpractice>.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, VIVEK SRIVASTAVA can be reached on 571-272-7304. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Application/Control Number: 14/181,112
Art Unit: 2449

Page 20

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/HERMON ASRES/
Examiner, Art Unit 2449

/VIVEK SRIVASTAVA/
Supervisory Patent Examiner, Art Unit 2449



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
 Address: COMMISSIONER FOR PATENTS
 P.O. Box 1450
 Alexandria, Virginia 22313-1450
 www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
14/181,112	02/14/2014	Zvi Tannenbaum	ZTANN.002P1C3	6874
20995 7590 05/22/2018 KNOBBE MARTENS OLSON & BEAR LLP 2040 MAIN STREET FOURTEENTH FLOOR IRVINE, CA 92614 UNITED STATES OF AMERICA			EXAMINER ASRES, HERMON	
			ART UNIT	PAPER NUMBER
			2449	
			NOTIFICATION DATE	DELIVERY MODE
			05/22/2018	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

jayna.cartee@knobbe.com
 efiling@knobbe.com

<i>Applicant-Initiated Interview Summary</i>	Application No. 14/181,112	Applicant(s) TANNENBAUM ET AL.	
	Examiner HERMON ASRES	Art Unit 2449	

All participants (applicant, applicant's representative, PTO personnel):

(1) HERMON ASRES. (3) Lance Smemoe (REG NO 66,152).

(2) _____. (4) _____.

Date of Interview: 15 May 2018.

Type: ☐ Telephonic ☐ Video Conference
☒ Personal [copy given to: ☐ applicant ☐ applicant's representative]

Exhibit shown or demonstration conducted: ☐ Yes ☒ No.
If Yes, brief description: _____.

Issues Discussed ☐101 ☐112 ☐102 ☒103 ☐Others
(For each of the checked box(es) above, please describe below the issue and detailed description of the discussion)

Claim(s) discussed: 1.

Identification of prior art discussed: _____.

Substance of Interview
(For each issue discussed, provide a detailed description and indicate if agreement was reached. Some topics may include: identification or clarification of a reference or a portion thereof, claim interpretation, proposed amendments, arguments of any applied references etc...)

Applicant discussed proposed amendment to independent claim 1 including "wherein the plurality of nodes comprises: a first node comprising at least one hardware processor configured to access a memory comprising program code for a user interface and program code for a single-node kernel configured to interpret user code and distribute computational instructions to at least one of a plurality of other nodes for execution; a second node comprising at least one hardware processor with a plurality of processing cores, wherein the second node is configured to receive computational instructions from the first node, execute the instructions, and communicate a result of the computation to at least one other node; a third node comprising at least one processor with a plurality of processing cores, wherein the third node is configured to receive the result of an executed computation from another node, execute a second computation based in part on the received result, and communicate the result of the second computation to the first node". Examiner indicated that the proposed amendment when formally filed would overcome the cited references and also that a search will be updated. No other agreement was reached.

Applicant recordation instructions: The formal written reply to the last Office action must include the substance of the interview. (See MPEP section 713.04). If a reply to the last Office action has already been filed, applicant is given a non-extendable period of the longer of one month or thirty days from this interview date, or the mailing date of this interview summary form, whichever is later, to file a statement of the substance of the interview

Examiner recordation instructions: Examiners must summarize the substance of any interview of record. A complete and proper recordation of the substance of an interview should include the items listed in MPEP 713.04 for complete and proper recordation including the identification of the general thrust of each argument or issue discussed, a general indication of any other pertinent matters discussed regarding patentability and the general results or outcome of the interview, to include an indication as to whether or not agreement was reached on the issues raised.

☒ Attachment

/HERMON ASRES/ Examiner, Art Unit 2449	/VIVEK SRIVASTAVA/ Supervisory Patent Examiner, Art Unit 2449
---	--

Manual of Patent Examining Procedure (MPEP), Section 713.04, Substance of Interview Must be Made of Record

A complete written statement as to the substance of any face-to-face, video conference, or telephone interview with regard to an application must be made of record in the application whether or not an agreement with the examiner was reached at the interview.

Title 37 Code of Federal Regulations (CFR) § 1.133 Interviews
Paragraph (b)

In every instance where reconsideration is requested in view of an interview with an examiner, a complete written statement of the reasons presented at the interview as warranting favorable action must be filed by the applicant. An interview does not remove the necessity for reply to Office action as specified in §§ 1.111, 1.135. (35 U.S.C. 132)

37 CFR §1.2 Business to be transacted in writing.

All business with the Patent or Trademark Office should be transacted in writing. The personal attendance of applicants or their attorneys or agents at the Patent and Trademark Office is unnecessary. The action of the Patent and Trademark Office will be based exclusively on the written record in the Office. No attention will be paid to any alleged oral promise, stipulation, or understanding in relation to which there is disagreement or doubt.

The action of the Patent and Trademark Office cannot be based exclusively on the written record in the Office if that record is itself incomplete through the failure to record the substance of interviews.

It is the responsibility of the applicant or the attorney or agent to make the substance of an interview of record in the application file, unless the examiner indicates he or she will do so. It is the examiner's responsibility to see that such a record is made and to correct material inaccuracies which bear directly on the question of patentability.

Examiners must complete an Interview Summary Form for each interview held where a matter of substance has been discussed during the interview by checking the appropriate boxes and filling in the blanks. Discussions regarding only procedural matters, directed solely to restriction requirements for which interview recordation is otherwise provided for in Section 812.01 of the Manual of Patent Examining Procedure, or pointing out typographical errors or unreadable script in Office actions or the like, are excluded from the interview recordation procedures below. Where the substance of an interview is completely recorded in an Examiners Amendment, no separate Interview Summary Record is required.

The Interview Summary Form shall be given an appropriate Paper No., placed in the right hand portion of the file, and listed on the "Contents" section of the file wrapper. In a personal interview, a duplicate of the Form is given to the applicant (or attorney or agent) at the conclusion of the interview. In the case of a telephone or video-conference interview, the copy is mailed to the applicant's correspondence address either with or prior to the next official communication. If additional correspondence from the examiner is not likely before an allowance or if other circumstances dictate, the Form should be mailed promptly after the interview rather than with the next official communication.

The Form provides for recordation of the following information:

- Application Number (Series Code and Serial Number)
- Name of applicant
- Name of examiner
- Date of interview
- Type of interview (telephonic, video-conference, or personal)
- Name of participant(s) (applicant, attorney or agent, examiner, other PTO personnel, etc.)
- An indication whether or not an exhibit was shown or a demonstration conducted
- An identification of the specific prior art discussed
- An indication whether an agreement was reached and if so, a description of the general nature of the agreement (may be by attachment of a copy of amendments or claims agreed as being allowable). Note: Agreement as to allowability is tentative and does not restrict further action by the examiner to the contrary.
- The signature of the examiner who conducted the interview (if Form is not an attachment to a signed Office action)

It is desirable that the examiner orally remind the applicant of his or her obligation to record the substance of the interview of each case. It should be noted, however, that the Interview Summary Form will not normally be considered a complete and proper recordation of the interview unless it includes, or is supplemented by the applicant or the examiner to include, all of the applicable items required below concerning the substance of the interview.

A complete and proper recordation of the substance of any interview should include at least the following applicable items:

- 1) A brief description of the nature of any exhibit shown or any demonstration conducted,
- 2) an identification of the claims discussed,
- 3) an identification of the specific prior art discussed,
- 4) an identification of the principal proposed amendments of a substantive nature discussed, unless these are already described on the Interview Summary Form completed by the Examiner,
- 5) a brief identification of the general thrust of the principal arguments presented to the examiner,
(The identification of arguments need not be lengthy or elaborate. A verbatim or highly detailed description of the arguments is not required. The identification of the arguments is sufficient if the general nature or thrust of the principal arguments made to the examiner can be understood in the context of the application file. Of course, the applicant may desire to emphasize and fully describe those arguments which he or she feels were or might be persuasive to the examiner.)
- 6) a general indication of any other pertinent matters discussed, and
- 7) if appropriate, the general results or outcome of the interview unless already described in the Interview Summary Form completed by the examiner.

Examiners are expected to carefully review the applicant's record of the substance of an interview. If the record is not complete and accurate, the examiner will give the applicant an extendable one month time period to correct the record.

Examiner to Check for Accuracy

If the claims are allowable for other reasons of record, the examiner should send a letter setting forth the examiner's version of the statement attributed to him or her. If the record is complete and accurate, the examiner should place the indication, "Interview Record OK" on the paper recording the substance of the interview along with the date and the examiner's initials.

<i>Applicant-Initiated Interview Summary</i>	Application No. 14/181,112	Applicant(s) TANNENBAUM ET AL.	
	Examiner HERMON ASRES	Art Unit 2449	

All participants (applicant, applicant's representative, PTO personnel):

(1) HERMON ASRES. (3) Lance Smemoe (REG NO 66,152).

(2) _____. (4) _____.

Date of Interview: 15 May 2018.

Type: ☐ Telephonic ☐ Video Conference
☒ Personal [copy given to: ☐ applicant ☐ applicant's representative]

Exhibit shown or demonstration conducted: ☐ Yes ☒ No.
If Yes, brief description: _____.

Issues Discussed ☐101 ☐112 ☐102 ☒103 ☐Others
(For each of the checked box(es) above, please describe below the issue and detailed description of the discussion)

Claim(s) discussed: 1.

Identification of prior art discussed: _____.

Substance of Interview
(For each issue discussed, provide a detailed description and indicate if agreement was reached. Some topics may include: identification or clarification of a reference or a portion thereof, claim interpretation, proposed amendments, arguments of any applied references etc...)

Applicant discussed proposed amendment to independent claim 1 including "wherein the plurality of nodes comprises: a first node comprising at least one hardware processor configured to access a memory comprising program code for a user interface and program code for a single-node kernel configured to interpret user code and distribute computational instructions to at least one of a plurality of other nodes for execution; a second node comprising at least one hardware processor with a plurality of processing cores, wherein the second node is configured to receive computational instructions from the first node, execute the instructions, and communicate a result of the computation to at least one other node; a third node comprising at least one processor with a plurality of processing cores, wherein the third node is configured to receive the result of an executed computation from another node, execute a second computation based in part on the received result, and communicate the result of the second computation to the first node". Examiner indicated that the proposed amendment when formally filed would overcome the cited references and also that a search will be updated. No other agreement was reached.

Applicant recordation instructions: The formal written reply to the last Office action must include the substance of the interview. (See MPEP section 713.04). If a reply to the last Office action has already been filed, applicant is given a non-extendable period of the longer of one month or thirty days from this interview date, or the mailing date of this interview summary form, whichever is later, to file a statement of the substance of the interview

Examiner recordation instructions: Examiners must summarize the substance of any interview of record. A complete and proper recordation of the substance of an interview should include the items listed in MPEP 713.04 for complete and proper recordation including the identification of the general thrust of each argument or issue discussed, a general indication of any other pertinent matters discussed regarding patentability and the general results or outcome of the interview, to include an indication as to whether or not agreement was reached on the issues raised.

☒ Attachment

/HERMON ASRES/ Examiner, Art Unit 2449	/VIVEK SRIVASTAVA/ Supervisory Patent Examiner, Art Unit 2449
---	--

Manual of Patent Examining Procedure (MPEP), Section 713.04, Substance of Interview Must be Made of Record

A complete written statement as to the substance of any face-to-face, video conference, or telephone interview with regard to an application must be made of record in the application whether or not an agreement with the examiner was reached at the interview.

Title 37 Code of Federal Regulations (CFR) § 1.133 Interviews
Paragraph (b)

In every instance where reconsideration is requested in view of an interview with an examiner, a complete written statement of the reasons presented at the interview as warranting favorable action must be filed by the applicant. An interview does not remove the necessity for reply to Office action as specified in §§ 1.111, 1.135. (35 U.S.C. 132)

37 CFR §1.2 Business to be transacted in writing.

All business with the Patent or Trademark Office should be transacted in writing. The personal attendance of applicants or their attorneys or agents at the Patent and Trademark Office is unnecessary. The action of the Patent and Trademark Office will be based exclusively on the written record in the Office. No attention will be paid to any alleged oral promise, stipulation, or understanding in relation to which there is disagreement or doubt.

The action of the Patent and Trademark Office cannot be based exclusively on the written record in the Office if that record is itself incomplete through the failure to record the substance of interviews.

It is the responsibility of the applicant or the attorney or agent to make the substance of an interview of record in the application file, unless the examiner indicates he or she will do so. It is the examiner's responsibility to see that such a record is made and to correct material inaccuracies which bear directly on the question of patentability.

Examiners must complete an Interview Summary Form for each interview held where a matter of substance has been discussed during the interview by checking the appropriate boxes and filling in the blanks. Discussions regarding only procedural matters, directed solely to restriction requirements for which interview recordation is otherwise provided for in Section 812.01 of the Manual of Patent Examining Procedure, or pointing out typographical errors or unreadable script in Office actions or the like, are excluded from the interview recordation procedures below. Where the substance of an interview is completely recorded in an Examiners Amendment, no separate Interview Summary Record is required.

The Interview Summary Form shall be given an appropriate Paper No., placed in the right hand portion of the file, and listed on the "Contents" section of the file wrapper. In a personal interview, a duplicate of the Form is given to the applicant (or attorney or agent) at the conclusion of the interview. In the case of a telephone or video-conference interview, the copy is mailed to the applicant's correspondence address either with or prior to the next official communication. If additional correspondence from the examiner is not likely before an allowance or if other circumstances dictate, the Form should be mailed promptly after the interview rather than with the next official communication.

The Form provides for recordation of the following information:

- Application Number (Series Code and Serial Number)
- Name of applicant
- Name of examiner
- Date of interview
- Type of interview (telephonic, video-conference, or personal)
- Name of participant(s) (applicant, attorney or agent, examiner, other PTO personnel, etc.)
- An indication whether or not an exhibit was shown or a demonstration conducted
- An identification of the specific prior art discussed
- An indication whether an agreement was reached and if so, a description of the general nature of the agreement (may be by attachment of a copy of amendments or claims agreed as being allowable). Note: Agreement as to allowability is tentative and does not restrict further action by the examiner to the contrary.
- The signature of the examiner who conducted the interview (if Form is not an attachment to a signed Office action)

It is desirable that the examiner orally remind the applicant of his or her obligation to record the substance of the interview of each case. It should be noted, however, that the Interview Summary Form will not normally be considered a complete and proper recordation of the interview unless it includes, or is supplemented by the applicant or the examiner to include, all of the applicable items required below concerning the substance of the interview.

A complete and proper recordation of the substance of any interview should include at least the following applicable items:

- 1) A brief description of the nature of any exhibit shown or any demonstration conducted,
- 2) an identification of the claims discussed,
- 3) an identification of the specific prior art discussed,
- 4) an identification of the principal proposed amendments of a substantive nature discussed, unless these are already described on the Interview Summary Form completed by the Examiner,
- 5) a brief identification of the general thrust of the principal arguments presented to the examiner,
(The identification of arguments need not be lengthy or elaborate. A verbatim or highly detailed description of the arguments is not required. The identification of the arguments is sufficient if the general nature or thrust of the principal arguments made to the examiner can be understood in the context of the application file. Of course, the applicant may desire to emphasize and fully describe those arguments which he or she feels were or might be persuasive to the examiner.)
- 6) a general indication of any other pertinent matters discussed, and
- 7) if appropriate, the general results or outcome of the interview unless already described in the Interview Summary Form completed by the examiner.

Examiners are expected to carefully review the applicant's record of the substance of an interview. If the record is not complete and accurate, the examiner will give the applicant an extendable one month time period to correct the record.

Examiner to Check for Accuracy

If the claims are allowable for other reasons of record, the examiner should send a letter setting forth the examiner's version of the statement attributed to him or her. If the record is complete and accurate, the examiner should place the indication, "Interview Record OK" on the paper recording the substance of the interview along with the date and the examiner's initials.

M E M O R A N D U M

TO: Examiner Hermon Asres
Direct: 571-272-4257
Email: Hermon.Asres@uspto.gov

FROM: Lance Smemoe, Reg. #66,152

RE: Agenda for in-person interview on May 15, 2018 at 2 p.m. Eastern time

Application No.: 14/181112

Attorney Docket No.: ZTANN.002P1C3

INFORMAL DOCUMENT FOR DISCUSSION ONLY – DO NOT ENTER

I. Discuss Applicant's disclosure and claimed inventions

II. Discuss references cited in Office Action

- A. Block et al. (US 2005/0021751)
- B. Singh (US 8,601,101)
- C. Howard et al. (US 2003/0195938)

III. Discuss proposed claim amendments

- 1. (Canceled)
- 2. **(Currently amended)** A computer cluster comprising:

a plurality of nodes comprising a hardware processor, wherein one or more of the nodes are configured to receive a command to start a cluster initialization process for the computer cluster, and wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that executes computational instructions, when executed, causes the hardware processor to interpret user instructions, to evaluate mathematical expressions, and to produce results of mathematical expression evaluation; and

a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture;

wherein the plurality of nodes comprises:

a first node comprising at least one hardware processor configured to access a memory comprising program code for a user interface and program code for a single-node kernel configured to interpret user code and distribute

M E M O R A N D U M

Page 2

computational instructions to at least one of a plurality of other nodes for execution;

a second node comprising at least one hardware processor with a plurality of processing cores, wherein the second node is configured to receive computational instructions from the first node, execute the instructions, and communicate a result of the computation to at least one other node;

a third node comprising at least one processor with a plurality of processing cores, wherein the third node is configured to receive the result of an executed computation from another node, execute a second computation based in part on the received result , and communicate the result of the second computation to the first node.

~~wherein at least one of the nodes is configured to return at least one result of mathematical expression evaluation to a user interface or a script.~~

ZTANN.002P1C3

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

First Inventor	: Zvi Tannenbaum
App. No.	: 14/181112
Filed	: February 14, 2014
For	: CLUSTER COMPUTING
Examiner	: Asres, Hermon
Art Unit	: 2449
Conf. No.	: 6874

RESPONSE TO NON-FINAL OFFICE ACTION DATED APRIL 13, 2018

Mail Stop Amendment

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Commissioner:

In response to the Office Action dated April 13, 2018, please reconsider the rejections in view of the following remarks.

Amendments to the Claims are reflected in the listing of claims which begins on page 2.

Summary of Interview begins on page 12.

Remarks/Arguments begin on page 13.

Application No.: 14/181112
Filing Date: February 14, 2014

AMENDMENTS TO THE CLAIMS

1. (Canceled)
2. **(Currently amended)** A computer cluster comprising:
 - a plurality of nodes, wherein each of the plurality of nodes comprises comprising a hardware processor, wherein one or more of the nodes are configured to receive a command to start a cluster initialization process for the computer cluster, and wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, is capable of causing the hardware processor to evaluate mathematical expressions, when executed, causes the hardware processor to interpret user instructions, to evaluate mathematical expressions, and to produce results of mathematical expression evaluation; and
 - a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture;
 - wherein the plurality of nodes comprises:
 - a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and
 - a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of the first mathematical expression evaluation to a third node;
 - wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;

Application No.: 14/181112
Filing Date: February 14, 2014

wherein ~~at least one of the nodes~~ the first node is configured to return ~~at least one~~ the result of the second mathematical expression evaluation to ~~a user interface or a script~~ the user interface.

3. (Previously presented) The computer cluster of claim 2, wherein the single-node kernel comprises a Mathematica kernel.

4. (Previously presented) The computer cluster of claim 2, wherein the mechanism comprises a message passing interface.

5. (Previously presented) The computer cluster of claim 2, wherein each of the nodes comprises one or more cluster node modules.

6. (Previously presented) The computer cluster of claim 2, wherein each single-node kernel is stored in a non-transitory computer-readable medium and configured to accept and execute a request.

7. (Previously presented) The computer cluster of claim 6, wherein each of the nodes comprises one or more cluster node modules, wherein each of the cluster node modules comprises instructions stored in a non-transitory computer-readable medium, and wherein the instructions, when executed by the hardware processor, cause the cluster node module to communicate with the single-node kernel and with one or more other cluster node modules.

8. (Previously presented) The computer cluster of claim 7, wherein the plurality of cluster node modules act as a cluster.

9. (Previously presented) The computer cluster of claim 7, wherein the plurality of cluster node modules communicate with one another to act as a cluster.

10. (Previously presented) The computer cluster of claim 9, wherein the computer cluster includes the user interface.

11. (Previously presented) The computer cluster of claim 10, wherein each cluster node module accepts instructions from the user interface and interprets one or more of the instructions.

12. (Previously presented) The computer cluster of claim 2, wherein the cluster initialization process comprises establishing communication among two or more of the nodes.

Application No.: 14/181112
Filing Date: February 14, 2014

13. (Previously presented) The computer cluster of claim 2, wherein the cluster initialization process comprises configuring access by one or more of the nodes to a computer-readable medium comprising program code for the single-node kernel.

14. (Previously presented) The computer cluster of claim 13, wherein the cluster initialization process comprises establishing message-passing support among the nodes in the cluster.

15. (Previously presented) The computer cluster of claim 13, wherein the cluster initialization process comprises launching cluster node modules by the cluster.

16. (Previously presented) The computer cluster of claim 15, wherein the cluster initialization process comprises assigning a processor identification number to each of the cluster node modules.

17. (Previously presented) The computer cluster of claim 2, wherein the cluster initialization process comprises:

launching cluster node modules by the cluster; and

after launching the cluster node modules, configuring access by one or more of the nodes to a non-transitory computer-readable medium comprising program code for the single-node kernel.

18. (Previously presented) The computer cluster of claim 2, wherein the cluster initialization process comprises:

launching cluster node modules by the cluster;

after launching the cluster node modules, establishing communication among two or more of the nodes; and

after establishing communication among two or more of the nodes, assigning a processor identification number to each of the cluster node modules.

19. (Previously presented) The computer cluster of claim 2, wherein one or more of the nodes are configured to communicate at least some of the user instructions.

20. (Previously presented) The computer cluster of claim 19, wherein one or more of the nodes are configured to communicate at least some of the user instructions to one or more single-node kernels.

Application No.: 14/181112
Filing Date: February 14, 2014

21. (Previously presented) The computer cluster of claim 2, wherein one or more of the nodes are configured to accept user instructions via one or more of the nodes.

22. (Previously presented) The computer cluster of claim 21, wherein one or more of the nodes are configured to communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other.

23. (Previously presented) The computer cluster of claim 2, wherein one or more of the nodes are configured to transmit at least some of the user instructions that originate from a user interface.

24. (Previously presented) The computer cluster of claim 2, wherein one or more of the nodes are configured to parallelize at least some of the user instructions before communicating at least some of the user instructions to one or more single-node kernels.

25. (Previously presented) The computer cluster of claim 2, wherein one or more of the nodes are configured to accept user instructions before communicating at least some of the user instructions to one or more single-node kernels.

26. (Previously presented) The computer cluster of claim 2, wherein one or more of the nodes are configured to:

accept user instructions;

after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and

after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to one or more single-node kernels.

27. (Previously presented) The computer cluster of claim 2, wherein the plurality of nodes are configured to communicate with one another to interpret and translate commands for execution by a plurality of single-node kernels.

28. **(Currently amended)** A computer cluster comprising:

a plurality of nodes, wherein one or more of the nodes are configured to receive:

a command to start a cluster initialization process for the computer cluster, wherein the cluster initialization process comprises establishing communication among two or more of the nodes; and

an instruction from a user interface or a script;

Application No.: 14/181112
Filing Date: February 14, 2014

and

a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using asynchronous calls;

wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, is capable of causing a hardware processor to evaluate mathematical expressions configured to interpret user instructions, to evaluate mathematical expressions, and to produce results of mathematical expression evaluation;[[and]]

wherein the plurality of nodes comprises:

a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and

a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of mathematical expression evaluation to a third node;

wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;

wherein ~~at least one of the nodes~~ the first node is configured to return ~~at least one~~ the result of the second mathematical expression evaluation to the user interface or the script.

29. (Previously presented) The computer cluster of claim 28, wherein the asynchronous calls comprise a first command to create a first packet containing:

an expression to be sent as payload; and

Application No.: 14/181112
Filing Date: February 14, 2014

a target node where the expression should be sent;
wherein the first command is configured to be called from within a single-node kernel;
wherein the single-node kernel is configured to send the first packet to a local cluster node module; and
wherein the local cluster node module is configured to forward the expression to the target node.

30. (Previously presented) The computer cluster of claim 29, wherein the asynchronous calls comprise a second command to create a second packet containing:

a location where the expression is expected to be received; and
a sending node from which the expression is expected to be received;
wherein the second command is configured to be called from within a single-node kernel;
wherein the single-node kernel is configured to send the second packet to a local cluster node module; and
wherein the local cluster node module is configured to store the second packet contents in a message receiving queue.

31. **(Currently amended)** A computer cluster comprising:

a plurality of nodes, wherein one or more of the nodes are configured to receive:
a command to start a cluster initialization process for the computer cluster, wherein the cluster initialization process comprises establishing communication among two or more of the nodes; and
an instruction from a user interface or a script;
and
a mechanism for the nodes to communicate results of mathematical expression evaluation with each other;
wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, is capable of causing a hardware processor to evaluate mathematical expressions

Application No.: 14/181112
Filing Date: February 14, 2014

~~configured to interpret user instructions, to evaluate mathematical expressions, and to produce results of mathematical expression evaluation;[[and]]~~

wherein the plurality of nodes comprises:

a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and

a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of mathematical expression evaluation to a third node;

wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;

and

~~wherein at least one of the nodes~~ the first node ~~is configured to return at least one the result of the second mathematical expression evaluation to the user interface or the script.~~

32. (Previously presented) The computer cluster of claim 5, wherein each of the plurality of nodes implements asynchronous calls that enable the single-node kernel to perform computation tasks while the cluster node modules are simultaneously communicating with one another.

33. (Previously presented) The computer cluster of claim 5, wherein intercommunication among the plurality of single-node kernels during thread execution is enabled by the plurality of cluster node modules, and wherein the computer cluster is configured to permit exchange of information between nodes during the course of a parallel computation.

Application No.: 14/181112
Filing Date: February 14, 2014

34. (Previously presented) The computer cluster of claim 5, wherein each of the single-node kernels are configured to call a send command involving creating a packet containing:

an expression to be sent as payload; and

where the expression should be sent;

wherein, upon reception of the send command by a local cluster node module associated with the single-node kernel, the local cluster node module is configured to decode the packet and forward a payload to the cluster node module specified in the packet;

wherein each of the single-node kernels is configured to call a receipt command involving creating a packet specifying which message to test for completion and to then wait for a reply expression to evaluate;

wherein, upon reception of the receipt command by the local cluster node module associated with the single-node kernel, the local cluster node module is configured to decode the packet and use a message specifier to search for any matching expressions listed as completed in a received message queue;

wherein, upon determining that such a completed expression is found, the local cluster node module is configured to send the completed expression to a local single-node kernel associated with the cluster node module in response to the receipt command, and

wherein each of the single-node kernels is configured to receive the completed expression and to update variables used by the single-node kernel during evaluation of expressions.

35. (Previously presented) The computer cluster of claim 2, wherein the plurality of nodes are configured to permit exchange of information between nodes during the course of parallel computation.

36. (Previously presented) The computer cluster of claim 2, wherein each of the plurality of nodes comprises instructions executable by the hardware processor and configured to implement asynchronous behavior, wherein the instructions comprise:

a first instruction to asynchronously send a payload to another node;

a second instruction to asynchronously receive a payload from another node; and

Application No.: 14/181112
Filing Date: February 14, 2014

a third instruction to search for a payload matching a message specifier.

37. **(Currently amended)** A computer cluster node for evaluating expressions in parallel with other computer cluster nodes, the computer cluster node comprising:

a hardware processor configured to access one or more non-transitory memory devices comprising program code for a single-node kernel that, when executed, causes the hardware processor to interpret user instructions, to evaluate mathematical expressions, and to produce results of mathematical expression evaluation;

a user connection interface configured to receive a command to start a cluster initialization process for a computer cluster;[[and]]

a mechanism to communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture; and

program code that, when executed, is capable of causing the hardware processor to receive calls from a first node, execute at least a first mathematical expression evaluation, and communicate a result of mathematical expression evaluation to another node for use in at least a second mathematical expression evaluation;

wherein the user connection interface is configured to return at least one result of mathematical expression evaluation to a user interface or a script.

38. (Previously presented) The computer cluster node of claim 37, wherein the one or more non-transitory memory devices comprise program code for performing a parallel fast Fourier transform, wherein the program code causes the hardware processor to evaluate a command to perform a Fourier transform on an array comprising a first data portion that is stored on the computer cluster node and a second data portion that is not stored on the computer cluster node.

39. (Previously presented) The computer cluster node of claim 37, wherein the computer cluster node is configured to permit exchange of information with other computer cluster nodes during the course of parallel computation.

40. (Previously presented) The computer cluster node of claim 37, wherein the computer cluster node comprises instructions executable by the hardware processor and configured to implement asynchronous behavior, wherein the instructions comprise:

a first instruction to asynchronously send a payload to another node;

Application No.: 14/181112

Filing Date: February 14, 2014

a second instruction to asynchronously receive a payload from another node; and

a third instruction to search for a payload matching a message specifier.

41. (Previously presented) The computer cluster node of claim 37, wherein the hardware processor comprises a special purpose microprocessor.

Application No.: 14/181112
Filing Date: February 14, 2014

SUMMARY OF INTERVIEW

Attendees, Date and Type of Interview

An in-person interview was conducted on May 15, 2018 and attended by the Examiner, Mr. Hermon Asres and the Applicant's representative, Mr. Lance D. Smemoe (Reg. No. 66,152).

Exhibits and/or Demonstrations

None

Identification of Claims Discussed

Claim 1

Identification of Cited/Disclosed Art Discussed

The art of record, including Block (US 2005/0021751), Singh (US 8,601,101), and Howard (US 2003/0195938), was discussed.

Proposed Amendments

The Applicant's representative presented a proposed amendment substantially as set forth in claim 1 as amended herein.

Principal Arguments and Other Matters

While reserving the right to pursue the subject matter as previously presented in the claims, the Applicant's representative argued that the claims as amended distinguished the references cited in the outstanding office action.

Results of Interview

The Examiner indicated that the proposed amendment when formally filed would overcome the cited references. The Applicant's representative will file a response, and the Examiner will reconsider the allowability of the claims after conducting an updated search.

Application No.: 14/181112
Filing Date: February 14, 2014

REMARKS

In the Office Action dated April 13, 2018, Claims 2-41 were rejected, as discussed below. In this Amendment, claims 2, 28, 31, and 37 are amended. No new matter is added. After the amendments, claims 2-41 remain pending for consideration. Claims 2, 28, 31, and 37 are independent claims.

In-person Interview

Applicant thanks the Examiner for the courteous and helpful in-person interview held on May 15, 2018 (summarized above).

Response to Double Patenting Rejection

Claims 2, 28, and 31 are rejected on the ground of non-statutory (obviousness-type) double patenting over claims 1 and 8 of U.S. Patent No. 8,676,877. Applicant respectfully requests that the double patenting rejections be held in abeyance until the other rejections are overcome.

Response to Rejections under 35 U.S.C. § 103

The Office Action rejects Claims 2 and 4-41 under 35 U.S.C. § 103, arguing that they are unpatentable in view of U.S. Publication No. 2005/0021751 to Block in view of U.S. Patent No. 8,601,101 to Singh and further in view of U.S. Publication No. 2003/0195938 to Howard. The Office Action rejects Claim 3 under 35 U.S.C. § 103, arguing that it is unpatentable in view of U.S. Publication No. 2005/0021751 to Block, U.S. Patent No. 8,601,101 to Singh, U.S. Publication No. 2007/0073705 to Gray, and U.S. Publication 2003/0195938 to Howard. Applicant respectfully traverses the rejections.

Applicant has amended the independent claims in an effort to advance prosecution of the application. No new matter is added. The application as filed explains that a first node of a computer cluster can be connected to a user interface or a script and configured to receive user instructions from the user interface or the script. *E.g.*, application as filed at ¶¶ [0012], [0025]. The first node can distribute calls to other cluster nodes. *E.g.*, *id.* at ¶ [0026]. The cluster nodes can evaluate expressions according to a set of rules. *E.g.*, *id.* at ¶ [0079]. The application

Application No.: 14/181112
Filing Date: February 14, 2014

describes an intercommunication architecture and functions that permit intermediate results to be passed from one node to another node without involvement of the first node. *E.g., id.* at ¶¶ [0027], [0121]. For example, in a non-limiting embodiment, the application describes a function that allows elements near the edge of a list, which can include intermediate results of evaluation, to be copied to neighboring processors so that neighboring edges of each node partition can interact. *Id.* at ¶¶ [0080], [0137]. Results of evaluations can be communicated back to the first node and returned to the user interface or script. *E.g., id.* at ¶¶ [0026], [0133].

As explained in the interview of May 15, 2018, the cited references do not disclose the combination of features recited in the claims as amended. For example, the references even if combined do not disclose a first node configured to interpret user instructions and distribute calls to other nodes for execution; a second node configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result to a third node; and a third node configured to receive the result, execute at least a second mathematical expression evaluation using the result, and communicate the result of the second mathematical expression evaluation to the first node, in combination with the other features recited in Claim 2.

Like Claim 2, independent Claims 28, 31, and 37 similarly recite subject matter not anticipated or rendered obvious by the combination of cited references. The dependent claims are patentable because each depends directly or indirectly from a patentable independent claim and because of the additional features recited in each claim. For at least these reasons, Applicant requests withdrawal of the rejections under 35 U.S.C. § 103.

No Disclaimers or Disavowals

Applicant respectfully submits that the claims are in condition for allowance. Furthermore, any remarks in support of patentability of one claim should not be imputed to any other claim, even if similar terminology is used. Any remarks referring to only a portion of a claim should not be understood to base patentability on that portion or that the limitation discussed is essential or critical; rather, patentability must rest on each claim taken as a whole. Applicant respectfully traverses each of the Examiner's rejections and each of the Examiner's assertions regarding what the prior art shows or teaches, even if not expressly discussed herein. Although the present communication may include alterations to the application or claims, or

Application No.: 14/181112
Filing Date: February 14, 2014

characterizations of claim scope or referenced art, applicant is not conceding in this application that previously pending claims are not patentable over the cited references. Rather, any alterations or characterizations are being made to facilitate expeditious prosecution of this application. Applicant reserves the right to pursue at a later date any previously pending or other broader or narrower claims that capture any subject matter supported by the present disclosure, including subject matter found to be specifically disclaimed herein or by any prior prosecution. Accordingly, reviewers of this or any parent, child or related prosecution history shall not reasonably infer that applicant has made any disclaimers or disavowals of any subject matter supported by the present application.

CONCLUSION

For the foregoing reasons, it is respectfully submitted that the rejections set forth in the outstanding Office Action are inapplicable to the present claims. Accordingly, issuance of a Notice of Allowance is requested.

The undersigned has made a good faith effort to respond to all of the rejections in the case and to place the claims in condition for immediate allowance. Nevertheless, if any undeveloped issues remain or if any issues require clarification, the Examiner is respectfully requested to call the undersigned at 949-721-5280.

Please charge any additional fees, including any fees for additional extension of time, or credit overpayment to Deposit Account No. 11-1410.

Respectfully submitted,

KNOBBE, MARTENS, OLSON & BEAR, LLP

Dated: October 5, 2018

By: /Lance D. Smemoe/

Lance D. Smemoe
Registration No. 66,152
Registered Practitioner
Customer No. 20995
(949) 760-0404



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
 United States Patent and Trademark Office
 Address: COMMISSIONER FOR PATENTS
 P.O. Box 1450
 Alexandria, Virginia 22313-1450
 www.uspto.gov

NOTICE OF ALLOWANCE AND FEE(S) DUE

20995 7590 11/29/2018
 KNOBBE MARTENS OLSON & BEAR LLP
 2040 MAIN STREET
 FOURTEENTH FLOOR
 IRVINE, CA 92614

EXAMINER	
ASRES, HERMON	
ART UNIT	PAPER NUMBER
2449	

DATE MAILED: 11/29/2018

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
14/181,112	02/14/2014	Zvi Tannenbaum	ZTANN.002P1C3	6874

TITLE OF INVENTION: CLUSTER COMPUTING

APPLN. TYPE	ENTITY STATUS	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	SMALL	\$500	\$0.00	\$0.00	\$500	02/28/2019

THE APPLICATION IDENTIFIED ABOVE HAS BEEN EXAMINED AND IS ALLOWED FOR ISSUANCE AS A PATENT. PROSECUTION ON THE MERITS IS CLOSED. THIS NOTICE OF ALLOWANCE IS NOT A GRANT OF PATENT RIGHTS. THIS APPLICATION IS SUBJECT TO WITHDRAWAL FROM ISSUE AT THE INITIATIVE OF THE OFFICE OR UPON PETITION BY THE APPLICANT. SEE 37 CFR 1.313 AND MPEP 1308.

THE ISSUE FEE AND PUBLICATION FEE (IF REQUIRED) MUST BE PAID WITHIN THREE MONTHS FROM THE MAILING DATE OF THIS NOTICE OR THIS APPLICATION SHALL BE REGARDED AS ABANDONED. THIS STATUTORY PERIOD CANNOT BE EXTENDED. SEE 35 U.S.C. 151. THE ISSUE FEE DUE INDICATED ABOVE DOES NOT REFLECT A CREDIT FOR ANY PREVIOUSLY PAID ISSUE FEE IN THIS APPLICATION. IF AN ISSUE FEE HAS PREVIOUSLY BEEN PAID IN THIS APPLICATION (AS SHOWN ABOVE), THE RETURN OF PART B OF THIS FORM WILL BE CONSIDERED A REQUEST TO REAPPLY THE PREVIOUSLY PAID ISSUE FEE TOWARD THE ISSUE FEE NOW DUE.

HOW TO REPLY TO THIS NOTICE:

I. Review the ENTITY STATUS shown above. If the ENTITY STATUS is shown as SMALL or MICRO, verify whether entitlement to that entity status still applies.

If the ENTITY STATUS is the same as shown above, pay the TOTAL FEE(S) DUE shown above.

If the ENTITY STATUS is changed from that shown above, on PART B - FEE(S) TRANSMITTAL, complete section number 5 titled "Change in Entity Status (from status indicated above)".

For purposes of this notice, small entity fees are 1/2 the amount of undiscounted fees, and micro entity fees are 1/2 the amount of small entity fees.

II. PART B - FEE(S) TRANSMITTAL, or its equivalent, must be completed and returned to the United States Patent and Trademark Office (USPTO) with your ISSUE FEE and PUBLICATION FEE (if required). If you are charging the fee(s) to your deposit account, section "4b" of Part B - Fee(s) Transmittal should be completed and an extra copy of the form should be submitted. If an equivalent of Part B is filed, a request to reapply a previously paid issue fee must be clearly made, and delays in processing may occur due to the difficulty in recognizing the paper as an equivalent of Part B.

III. All communications regarding this application must give the application number. Please direct all communications prior to issuance to Mail Stop ISSUE FEE unless advised to the contrary.

IMPORTANT REMINDER: Maintenance fees are due in utility patents issuing on applications filed on or after Dec. 12, 1980. It is patentee's responsibility to ensure timely payment of maintenance fees when due. More information is available at www.uspto.gov/PatentMaintenanceFees.

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), by mail or fax, or via EFS-Web.

By mail, send to: Mail Stop ISSUE FEE
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450

By fax, send to: (571)-273-2885

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

20995 7590 11/29/2018
KNOBBE MARTENS OLSON & BEAR LLP
2040 MAIN STREET
FOURTEENTH FLOOR
IRVINE, CA 92614

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being transmitted to the USPTO via EFS-Web or by facsimile to (571) 273-2885, on the date below.

(Typed or printed name)
(Signature)
(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
14/181,112	02/14/2014	Zvi Tannenbaum	ZTANN.002P1C3	6874

TITLE OF INVENTION: CLUSTER COMPUTING

APPLN. TYPE	ENTITY STATUS	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	SMALL	\$500	\$0.00	\$0.00	\$500	02/28/2019

EXAMINER	ART UNIT	CLASS-SUBCLASS
ASRES, HERMON	2449	709-223000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).

☐ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.

☐ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-09 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list

(1) The names of up to 3 registered patent attorneys or agents OR, alternatively,

(2) The name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.

1 _____

2 _____

3 _____

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document must have been previously recorded, or filed for recordation, as set forth in 37 CFR 3.11 and 37 CFR 3.81(a). Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE

(B) RESIDENCE: (CITY and STATE OR COUNTRY)

Please check the appropriate assignee category or categories (will not be printed on the patent): ☐ Individual ☐ Corporation or other private group entity ☐ Government

4a. Fees submitted: ☐ Issue Fee ☐ Publication Fee (if required) ☐ Advance Order - # of Copies _____

4b. Method of Payment: (Please first reapply any previously paid fee shown above)

☐ Electronic Payment via EFS-Web ☐ Enclosed check ☐ Non-electronic payment by credit card (Attach form PTO-2038)

☐ The Director is hereby authorized to charge the required fee(s), any deficiency, or credit any overpayment to Deposit Account No. _____

5. **Change in Entity Status** (from status indicated above)

☐ Applicant certifying micro entity status. See 37 CFR 1.29

☐ Applicant asserting small entity status. See 37 CFR 1.27

☐ Applicant changing to regular undiscounted fee status.

NOTE: Absent a valid certification of Micro Entity Status (see forms PTO/SB/15A and 15B), issue fee payment in the micro entity amount will not be accepted at the risk of application abandonment.

NOTE: If the application was previously under micro entity status, checking this box will be taken to be a notification of loss of entitlement to micro entity status.

NOTE: Checking this box will be taken to be a notification of loss of entitlement to small or micro entity status, as applicable.

NOTE: This form must be signed in accordance with 37 CFR 1.31 and 1.33. See 37 CFR 1.4 for signature requirements and certifications.

Authorized Signature _____

Date _____

Typed or printed name _____

Registration No. _____



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
 Address: COMMISSIONER FOR PATENTS
 P.O. Box 1450
 Alexandria, Virginia 22313-1450
 www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
14/181,112	02/14/2014	Zvi Tannenbaum	ZTANN.002P1C3	6874
20995	7590	11/29/2018	EXAMINER	
KNOBBE MARTENS OLSON & BEAR LLP			ASRES, HERMON	
2040 MAIN STREET			ART UNIT	
FOURTEENTH FLOOR			PAPER NUMBER	
IRVINE, CA 92614			2449	
DATE MAILED: 11/29/2018				

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)
 (Applications filed on or after May 29, 2000)

The Office has discontinued providing a Patent Term Adjustment (PTA) calculation with the Notice of Allowance.

Section 1(h)(2) of the AIA Technical Corrections Act amended 35 U.S.C. 154(b)(3)(B)(i) to eliminate the requirement that the Office provide a patent term adjustment determination with the notice of allowance. See Revisions to Patent Term Adjustment, 78 Fed. Reg. 19416, 19417 (Apr. 1, 2013). Therefore, the Office is no longer providing an initial patent term adjustment determination with the notice of allowance. The Office will continue to provide a patent term adjustment determination with the Issue Notification Letter that is mailed to applicant approximately three weeks prior to the issue date of the patent, and will include the patent term adjustment on the patent. Any request for reconsideration of the patent term adjustment determination (or reinstatement of patent term adjustment) should follow the process outlined in 37 CFR 1.705.

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (571)-272-7702. Questions relating to issue and publication fee payments should be directed to the Customer Service Center of the Office of Patent Publication at 1-(888)-786-0101 or (571)-272-4200.

OMB Clearance and PRA Burden Statement for PTOL-85 Part B

The Paperwork Reduction Act (PRA) of 1995 requires Federal agencies to obtain Office of Management and Budget approval before requesting most types of information from the public. When OMB approves an agency request to collect information from the public, OMB (i) provides a valid OMB Control Number and expiration date for the agency to display on the instrument that will be used to collect the information and (ii) requires the agency to inform the public about the OMB Control Number's legal significance in accordance with 5 CFR 1320.5(b).

The information collected by PTOL-85 Part B is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 30 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450. Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C. 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Notice of Allowability	Application No. 14/181,112	Applicant(s) Tannenbaum et al.	
	Examiner HERMON ASRES	Art Unit 2449	AIA Status No

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to 10/05/2018.
☐ A declaration(s)/affidavit(s) under **37 CFR 1.130(b)** was/were filed on ____.
2. ☐ An election was made by the applicant in response to a restriction requirement set forth during the interview on ____; the restriction requirement and election have been incorporated into this action.
3. ☒ The allowed claim(s) is/are 2-25 and 27-41. As a result of the allowed claim(s), you may be eligible to benefit from the **Patent Prosecution Highway** program at a participating intellectual property office for the corresponding application. For more information, please see http://www.uspto.gov/patents/init_events/pph/index.jsp or send an inquiry to PPHfeedback@uspto.gov.
4. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

Certified copies:

a) ☐ All b) ☐ Some *c) ☐ None of the:

1. ☐ Certified copies of the priority documents have been received.
2. ☐ Certified copies of the priority documents have been received in Application No. ____.
3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

* Certified copies not received: ____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

5. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date ____.

Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).

6. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. <input checked="" type="checkbox"/> Notice of References Cited (PTO-892) 2. <input checked="" type="checkbox"/> Information Disclosure Statements (PTO/SB/08), Paper No./Mail Date ____. 3. <input type="checkbox"/> Examiner's Comment Regarding Requirement for Deposit of Biological Material ____. 4. <input checked="" type="checkbox"/> Interview Summary (PTO-413), Paper No./Mail Date. <u>11/21/2018</u> .	5. <input checked="" type="checkbox"/> Examiner's Amendment/Comment 6. <input checked="" type="checkbox"/> Examiner's Statement of Reasons for Allowance 7. <input type="checkbox"/> Other _____.
--	---

/HERMON ASRES/
Examiner, Art Unit 2449

Application/Control Number: 14/181,112
Art Unit: 2449

Page 2

DETAILED ACTION

Claims 2-25, and 27-41 are allowed.

Response to Arguments

Double Patenting rejection

Examiner withdraws the double patenting rejection of claims 2, 28, and 31 in view of the Terminal Disclaimer filed and approved on 11/21/2018

EXAMINER'S AMENDMENT

An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

Authorization for this examiner's amendment was given in a telephone correspondence with Lance Smemoe (Reg. No. 66,152) on 11/21/2018.

The application has been amended as follows:

1. (Canceled)
2. **(Currently amended)** A computer cluster comprising:
a plurality of nodes, wherein each of the plurality of nodes comprises a hardware processor, wherein one or more of the nodes are configured to receive a command to start a cluster initialization process for the computer cluster, and wherein each of the nodes is

Application/Control Number: 14/181,112
 Art Unit: 2449

Page 3

configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, is capable of causing the hardware processor to evaluate mathematical expressions; and

a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture;

wherein the plurality of nodes comprises:

a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and

a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of the first mathematical expression evaluation to a third node;

wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;

wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface;

wherein one or more of the nodes are configured to:

accept user instructions;

after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other;
and

Application/Control Number: 14/181,112
Art Unit: 2449

Page 4

after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to one or more single-node kernels.

3. (Previously presented) The computer cluster of claim 2, wherein the single-node kernel comprises a Mathematica kernel.

4. (Previously presented) The computer cluster of claim 2, wherein the mechanism comprises a message passing interface.

5. (Previously presented) The computer cluster of claim 2, wherein each of the nodes comprises one or more cluster node modules.

6. (Previously presented) The computer cluster of claim 2, wherein each single-node kernel is stored in a non-transitory computer-readable medium and configured to accept and execute a request.

7. (Previously presented) The computer cluster of claim 6, wherein each of the nodes comprises one or more cluster node modules, wherein each of the cluster node modules comprises instructions stored in a non-transitory computer-readable medium, and wherein the instructions, when executed by the hardware processor, cause the cluster node module to communicate with the single-node kernel and with one or more other cluster node modules.

8. (Previously presented) The computer cluster of claim 7, wherein the plurality of cluster node modules act as a cluster.

9. (Previously presented) The computer cluster of claim 7, wherein the plurality of cluster node modules communicate with one another to act as a cluster.

10. (Previously presented) The computer cluster of claim 9, wherein the computer cluster includes the user interface.

11. (Previously presented) The computer cluster of claim 10, wherein each cluster node module accepts instructions from the user interface and interprets one or more of the instructions.

Application/Control Number: 14/181,112
Art Unit: 2449

Page 5

12. (Previously presented) The computer cluster of claim 2, wherein the cluster initialization process comprises establishing communication among two or more of the nodes.

13. (Previously presented) The computer cluster of claim 2, wherein the cluster initialization process comprises configuring access by one or more of the nodes to a computer-readable medium comprising program code for the single-node kernel.

14. (Previously presented) The computer cluster of claim 13, wherein the cluster initialization process comprises establishing message-passing support among the nodes in the cluster.

15. (Previously presented) The computer cluster of claim 13, wherein the cluster initialization process comprises launching cluster node modules by the cluster.

16. (Previously presented) The computer cluster of claim 15, wherein the cluster initialization process comprises assigning a processor identification number to each of the cluster node modules.

17. (Previously presented) The computer cluster of claim 2, wherein the cluster initialization process comprises:

launching cluster node modules by the cluster; and

after launching the cluster node modules, configuring access by one or more of the nodes to a non-transitory computer-readable medium comprising program code for the single-node kernel.

18. (Previously presented) The computer cluster of claim 2, wherein the cluster initialization process comprises:

launching cluster node modules by the cluster;

after launching the cluster node modules, establishing communication among two or more of the nodes; and

after establishing communication among two or more of the nodes, assigning a processor identification number to each of the cluster node modules.

Application/Control Number: 14/181,112
Art Unit: 2449

Page 6

19. (Previously presented) The computer cluster of claim 2, wherein one or more of the nodes are configured to communicate at least some of the user instructions.

20. (Previously presented) The computer cluster of claim 19, wherein one or more of the nodes are configured to communicate at least some of the user instructions to one or more single-node kernels.

21. (Previously presented) The computer cluster of claim 2, wherein one or more of the nodes are configured to accept user instructions via one or more of the nodes.

22. (Previously presented) The computer cluster of claim 21, wherein one or more of the nodes are configured to communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other.

23. (Previously presented) The computer cluster of claim 2, wherein one or more of the nodes are configured to transmit at least some of the user instructions that originate from a user interface.

24. (Previously presented) The computer cluster of claim 2, wherein one or more of the nodes are configured to parallelize at least some of the user instructions before communicating at least some of the user instructions to one or more single-node kernels.

25. (Previously presented) The computer cluster of claim 2, wherein one or more of the nodes are configured to accept user instructions before communicating at least some of the user instructions to one or more single-node kernels.

26. (Canceled)

27. (Previously presented) The computer cluster of claim 2, wherein the plurality of nodes are configured to communicate with one another to interpret and translate commands for execution by a plurality of single-node kernels.

28. **(Currently amended)** A computer cluster comprising:
a plurality of nodes, wherein one or more of the nodes are configured to receive:

Application/Control Number: 14/181,112
Art Unit: 2449

Page 7

a command to start a cluster initialization process for the computer cluster, wherein the cluster initialization process comprises establishing communication among two or more of the nodes; and

an instruction from a user interface or a script;

and

a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using asynchronous calls;

wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, is capable of causing a hardware processor to evaluate mathematical expressions;

wherein the plurality of nodes comprises:

a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and

a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of mathematical expression evaluation to a third node;

wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;

wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface or the script;

wherein one or more of the nodes are configured to:

accept user instructions;

Application/Control Number: 14/181,112
 Art Unit: 2449

Page 8

after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and

after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to one or more single-node kernels.

29. (Previously presented) The computer cluster of claim 28, wherein the asynchronous calls comprise a first command to create a first packet containing:

an expression to be sent as payload; and

a target node where the expression should be sent;

wherein the first command is configured to be called from within a single-node kernel;

wherein the single-node kernel is configured to send the first packet to a local cluster node module; and

wherein the local cluster node module is configured to forward the expression to the target node.

30. (Previously presented) The computer cluster of claim 29, wherein the asynchronous calls comprise a second command to create a second packet containing:

a location where the expression is expected to be received; and

a sending node from which the expression is expected to be received;

wherein the second command is configured to be called from within a single-node kernel;

wherein the single-node kernel is configured to send the second packet to a local cluster node module; and

wherein the local cluster node module is configured to store the second packet contents in a message receiving queue.

31. **(Currently amended)** A computer cluster comprising:

a plurality of nodes, wherein one or more of the nodes are configured to receive:

Application/Control Number: 14/181,112
Art Unit: 2449

Page 9

a command to start a cluster initialization process for the computer cluster, wherein the cluster initialization process comprises establishing communication among two or more of the nodes; and

an instruction from a user interface or a script;

and

a mechanism for the nodes to communicate results of mathematical expression evaluation with each other;

wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, is capable of causing a hardware processor to evaluate mathematical expressions;

wherein the plurality of nodes comprises:

a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and

a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of mathematical expression evaluation to a third node;

wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;

and

wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface or the script;

wherein one or more of the nodes are configured to:

accept user instructions;

Application/Control Number: 14/181,112
 Art Unit: 2449

Page 10

after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and

after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to one or more single-node kernels.

32. (Previously presented) The computer cluster of claim 5, wherein each of the plurality of nodes implements asynchronous calls that enable the single-node kernel to perform computation tasks while the cluster node modules are simultaneously communicating with one another.

33. (Previously presented) The computer cluster of claim 5, wherein intercommunication among the plurality of single-node kernels during thread execution is enabled by the plurality of cluster node modules, and wherein the computer cluster is configured to permit exchange of information between nodes during the course of a parallel computation.

34. (Previously presented) The computer cluster of claim 5, wherein each of the single-node kernels are configured to call a send command involving creating a packet containing:

an expression to be sent as payload; and

where the expression should be sent;

wherein, upon reception of the send command by a local cluster node module associated with the single-node kernel, the local cluster node module is configured to decode the packet and forward a payload to the cluster node module specified in the packet;

wherein each of the single-node kernels is configured to call a receipt command involving creating a packet specifying which message to test for completion and to then wait for a reply expression to evaluate;

wherein, upon reception of the receipt command by the local cluster node module associated with the single-node kernel, the local cluster node module is configured to decode the packet and use a message specifier to search for any matching expressions listed as completed in a received message queue;

Application/Control Number: 14/181,112
 Art Unit: 2449

Page 11

wherein, upon determining that such a completed expression is found, the local cluster node module is configured to send the completed expression to a local single-node kernel associated with the cluster node module in response to the receipt command, and

wherein each of the single-node kernels is configured to receive the completed expression and to update variables used by the single-node kernel during evaluation of expressions.

35. (Previously presented) The computer cluster of claim 2, wherein the plurality of nodes are configured to permit exchange of information between nodes during the course of parallel computation.

36. (Previously presented) The computer cluster of claim 2, wherein each of the plurality of nodes comprises instructions executable by the hardware processor and configured to implement asynchronous behavior, wherein the instructions comprise:

- a first instruction to asynchronously send a payload to another node;
- a second instruction to asynchronously receive a payload from another node; and
- a third instruction to search for a payload matching a message specifier.

37. **(Currently amended)** A computer cluster node for evaluating expressions in parallel with other computer cluster nodes, the computer cluster node comprising:

a hardware processor configured to access one or more non-transitory memory devices comprising program code for a single-node kernel that, when executed, causes the hardware processor to interpret user instructions, to evaluate mathematical expressions, and to produce results of mathematical expression evaluation, wherein the hardware processor comprises multiple processor cores;

a user connection interface configured to receive a command to start a cluster initialization process for a computer cluster;

a mechanism to communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture; and

~~program code that, when executed, is capable of causing the hardware processor to receive calls from a first node, execute at least a first mathematical expression evaluation,~~

Application/Control Number: 14/181,112
 Art Unit: 2449

Page 12

~~and communicate a result of mathematical expression evaluation to another node for use in at least a second mathematical expression evaluation;~~

program code that, when executed, is capable of causing the hardware processor to:
receive calls from a second node comprising a second hardware processor
configured to access a second memory comprising program code for a user
interface and program code for a second single-node kernel, the second single-node
kernel configured to interpret user instructions and distribute calls to at least one of
a plurality of other nodes for execution;

execute, using the hardware processor, at least a first mathematical
expression evaluation; and

communicate a result of the first mathematical expression evaluation to a
third node comprising a third hardware processor with a plurality of processing
cores, wherein the third node is configured to receive the result of mathematical
expression evaluation from the computer cluster node, execute at least a second
mathematical expression evaluation using the result of the first mathematical
expression evaluation, and communicate a result of the second mathematical
expression evaluation to the first node;

wherein the user connection interface is configured to return at least one result of mathematical expression evaluation to a user interface or a script; and

wherein the computer cluster node is configured to:

accept user instructions;

after accepting user instructions, communicate at least some of the user
instructions using the mechanism for the nodes to communicate with each other;
and

after communicating at least some of the user instructions using the
mechanism, communicate at least some of the user instructions to the single-node
kernel.

38. (Previously presented) The computer cluster node of claim 37, wherein the one or more non-transitory memory devices comprise program code for performing a parallel fast Fourier transform, wherein the program code causes the hardware processor to evaluate a command to

Application/Control Number: 14/181,112
Art Unit: 2449

Page 13

perform a Fourier transform on an array comprising a first data portion that is stored on the computer cluster node and a second data portion that is not stored on the computer cluster node.

39. (Previously presented) The computer cluster node of claim 37, wherein the computer cluster node is configured to permit exchange of information with other computer cluster nodes during the course of parallel computation.

40. (Previously presented) The computer cluster node of claim 37, wherein the computer cluster node comprises instructions executable by the hardware processor and configured to implement asynchronous behavior, wherein the instructions comprise:

- a first instruction to asynchronously send a payload to another node;
- a second instruction to asynchronously receive a payload from another node; and
- a third instruction to search for a payload matching a message specifier.

41. (Previously presented) The computer cluster node of claim 37, wherein the hardware processor comprises a special purpose microprocessor.

Reason for Allowance

The following is an examiner's statement of reasons for allowance.

Independent Claims 2 as amended distinguishes itself over the prior art due to the amended limitation in combination with the rest of the limitations. It is to be noted that it is the combination of all limitations that renders the claims allowable. Claims 3-25, and 27-41 are allowed based on the same reason(s).

Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

Application/Control Number: 14/181,112
Art Unit: 2449

Page 14

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to HERMON ASRES whose telephone number is (571)272-4257. The examiner can normally be reached on Monday to Friday 9AM to 5PM.

Examiner interviews are available via telephone, in-person, and video conferencing using a USPTO supplied web-based collaboration tool. To schedule an interview, applicant is encouraged to use the USPTO Automated Interview Request (AIR) at <http://www.uspto.gov/interviewpractice>.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Vivek Srivastava can be reached on (571)272-7304. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/HERMON ASRES/
Examiner, Art Unit 2449

<i>Examiner-Initiated Interview Summary</i>	Application No. 14/181,112		Applicant(s) Tannenbaum et al.	
	Examiner HERMON ASRES		Art Unit 2449	AIA Status No

All participants (applicant, applicant's representative, PTO personnel):

(1) HERMON ASRES. (3) Lance Smemoe (REG NO 66,152).

(2) _____. (4) _____.

Date of Interview: 21 November 2018.

Type: ☒ Telephonic ☐ Video Conference
☐ Personal [copy given to: ☐ applicant ☐ applicant's representative]

Exhibit shown or demonstration conducted: ☐ Yes ☒ No.
If Yes, brief description: _____.

Issues Discussed ☐ 101 ☐ 112 ☐ 102 ☐ 103 ☒ Others
(For each of the checked box(es) above, please describe below the issue and detailed description of the discussion)

Claim(s) discussed: 2-41.

Identification of prior art discussed: _____.

Substance of Interview
(For each issue discussed, provide a detailed description and indicate if agreement was reached. Some topics may include: identification or clarification of a reference or a portion thereof, claim interpretation, proposed amendments, arguments of any applied references etc...)

See Continuation Sheet.


Applicant recordation instructions: It is not necessary for applicant to provide a separate record of the substance of interview.

Examiner recordation instructions: Examiners must summarize the substance of any interview of record. A complete and proper recordation of the substance of an interview should include the items listed in MPEP 713.04 for complete and proper recordation including the identification of the general thrust of each argument or issue discussed, a general indication of any other pertinent matters discussed regarding patentability and the general results or outcome of the interview, to include an indication as to whether or not agreement was reached on the issues raised.

☐ Attachment

/HERMON ASRES/ Examiner, Art Unit 2449	
---	--

Continuation of Substance of Interview including description of the general nature of what was agreed to if an agreement was reached, or any other comments: Examiner contacted applicant and proposed incorporating dependent claim 26 into all the Independent claims and also re-write independent claim 37 to have all the limitations of that other independent claims and cancel dependent claim 26 in order to put the application in condition for allowance. Applicant's representative Lance Smemoe (REG NO 66,152) has authorized the cancellation of dependent claim 26 and incorporating it's limitation to all the independent claims and also authorized make independent claim 37 have all the limitations from the other independent claims. Applicant has filed Terminal Disclaimer to overcome the double patenting rejection.

<i>Index of Claims</i> 	Application/Control No. 14/181,112	Applicant(s)/Patent Under Reexamination Tannenbaum et al.
	Examiner HERMON ASRES	Art Unit 2449

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

CLAIMS										
<input type="checkbox"/> Claims renumbered in the same order as presented by applicant <input type="checkbox"/> CPA <input type="checkbox"/> T.D. <input type="checkbox"/> R.1.47										
CLAIM		DATE								
Final	Original	07/07/2016	03/01/2017	02/18/2018	11/22/2018					
	1	-	-	-	-					
	2	✓	✓	✓	=					
	3	✓	✓	✓	=					
	4	✓	✓	✓	=					
	5	✓	✓	✓	=					
	6	✓	✓	✓	=					
	7	✓	✓	✓	=					
	8	✓	✓	✓	=					
	9	✓	✓	✓	=					
	10	✓	✓	✓	=					
	11	✓	✓	✓	=					
	12		✓	✓	=					
	13		✓	✓	=					
	14		✓	✓	=					
	15		✓	✓	=					
	16		✓	✓	=					
	17		✓	✓	=					
	18		✓	✓	=					
	19		✓	✓	=					
	20		✓	✓	=					
	21		✓	✓	=					
	22		✓	✓	=					
	23		✓	✓	=					
	24		✓	✓	=					
	25		✓	✓	=					
	26		✓	✓	-					
	27		✓	✓	=					
	28		✓	✓	=					
	29		✓	✓	=					
	30		✓	✓	=					
	31		✓	✓	=					
	32			✓	=					
	33			✓	=					
	34			✓	=					
	35			✓	=					
	36			✓	=					
	37			✓	=					
	38			✓	=					
	39			✓	=					
	40			✓	=					
	41			✓	=					

EXHIBIT D

**IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE**

ADVANCED CLUSTER SYSTEMS,
INC.,

Plaintiff,

v.

NVIDIA CORPORATION, NVIDIA
SINGAPORE PTE. LTD., AND
NVIDIA INTERNATIONAL, INC.

Defendants.

Civil Action No. 1:19-cv-02032-CFC

**EXPERT REPORT OF JASWINDER PAL SINGH, PH. D. ON CLAIM
CONSTRUCTION**

I. INTRODUCTION

1. I, Jaswinder Pal Singh, Ph.D., have been retained by counsel for the plaintiff, Advanced Cluster Systems, Inc. (“ACS”), to provide technical assistance in this and related actions.

2. I understand that ACS is alleging that the defendants, NVIDIA Corp., NVIDIA Singapore Pte. Ltd., and NVIDIA International, Inc. (collectively, “Defendants” or “NVIDIA”) infringe U.S. Patent 10,333,768 (the “’768 Patent”). This declaration relates to the interpretation of certain claim limitations in the ’768 Patent. Specifically, I have been asked to consider the following claim terms:

- “peer-to-peer architecture”
- “cluster node module”
- “kernel”
- “single-node kernel”
- “wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node.”
- “a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture”
- “a mechanism to communicate results of evaluation with other computer cluster nodes using a peer-to peer architecture”
- “a mechanism for the nodes to communicate results of mathematical expression evaluation with each other”
- “a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using asynchronous calls”
- “the first node”
- “communicate a result of the second mathematical evaluation to the first node”
- “wherein the third node is configured to receive the result of mathematical expression evaluation from the computer cluster node, execute at least a second mathematical expression evaluation using the result of the first mathematical expression evaluation, and communicate a result of the second mathematical expression evaluation to the first node.”

II. BACKGROUND AND EXPERIENCE

3. I am currently a Professor of Computer Science at Princeton University, a position I have held for more than 15 years. Before becoming a Full Professor, I was an Assistant, and then Associate, Professor at Princeton.

4. I received a Bachelor's Degree in Electrical Engineering and Computer Science, *summa cum laude*, from Princeton in 1987. I received an M.S. Degree in Electrical Engineering in 1989 and a Ph.D. Degree in Electrical Engineering in 1993, both from Stanford University. For my Ph.D. thesis, I researched the boundary of applications and parallel computer systems, including programming models, and the interactions between software and hardware. My current *curriculum vitae* ("CV") is attached to this declaration as Exhibit A.

5. I joined the faculty at Princeton in 1995, where I served as an Assistant Professor in the Computer Science Department. My initial research area included parallel computer systems and applications with a focus on (i) scalable multiprocessing in both the shared address space and message passing paradigms, including studying the programming and performance tradeoffs between the two, (ii) understanding how shared address space abstractions might be supported using commodity parts, and how applications might be restructured for such systems, and (iii) the use of high-performance computing in biology, medicine, cosmology, and other areas.

6. From 1999 to 2010 I was the director of Princeton's "Program in Integrative Computer and Application Sciences" (PICASso), a multi-department, university-wide interdisciplinary program focused on scalable parallel and distributed computing at the boundary of computer science and a broad range of application sciences. Directing that program required me to develop and foster new interdisciplinary courses, new practice-oriented workshops in scalable computing based on needs gathered from multiple departments, new interdisciplinary seminar series, and programs in effective written and oral scientific communication. I oversaw the creation of courses for an interdisciplinary computational science curriculum, taught by faculty in different departments, in addition to parallel and distributed computing courses. From 2003 to the present, I have been on the Executive Committee at Princeton's Institute for Computational Science and Engineering (PICSSiE). That permanent institute came out of the PICASso program that I led, and the Institute has taken over PICASso's activities as well, including a permanent Graduate Certificate Program in Computational Science and Engineering at Princeton that I initially developed.

7. In addition to my concurrent work as a professor, beginning in 1999, I was a member of the Technical Advisory Board at NOAA Geophysical Fluid Dynamics Laboratory. In that role I aided in the shift from vector to parallel

computing, and in the procurement of large parallel computing and analysis infrastructures.

8. At Princeton, I have taught many courses in areas relevant to the technology disclosed in the '768 Patent. These include undergraduate and graduate courses in parallel computing, parallel architecture and programming, scalable infrastructure and services, and analysis of data. I taught several new interdisciplinary courses I created for the PICASso program regarding parallel and distributed computing. I also created and taught courses in Marketplace Design and at the boundary of technology, business, marketplaces, and economics, including one which has been dubbed the "CTO Course."

9. In the course of my work at Princeton, I supervised more than 15 students engaged in thesis research projects in support of Master of Science and Doctor of Philosophy degrees. Those projects have included topics such as new programming models for scalable parallel computing, applications of parallel computing and their interactions with systems, peer-to-peer publish-subscribe systems, and a comparison of programming models for high-performance parallel computing.

10. I am the author of a leading textbook on parallel computing, "Parallel Computer Architecture: A Hardware-Software Approach," by David E. Culler,

Jaswinder Pal Singh with Anoop Gupta, Morgan Kaufmann Publishers, 1999. I have also published more than 100 referenced conference and journal publications.

11. I am a member of the Association for Computing Machinery (“ACM”) and the Institute of Electrical and Electronics Engineers (“IEEE”).

12. I have been recognized for my research and other professional endeavors. I was honored with a Presidential Early Career Award for Scientists and Engineers (PECASE), 1997, awarded to twenty young scientists and engineers in the United States selected from all areas of science/engineering by the National Science Foundation, and the Sloan Research Fellowship, 1998, awarded to ten Computer Scientists nationwide by the Alfred Sloan Foundation. Also, my 2005 publication for the International Distributed Event-based Systems (“DEBS”) was in 2019 voted as the most influential paper of DEBS 2005.

13. My professional activities further include serving on corporate boards of directors, including the companies 8x8, Inc. and Hiro, PBC (formerly Blockstack, PBC).

14. I have accepted invitations to evaluate research and education programs and to evaluate proposals, including for the U.S. Government (NOAA’s procurement programs and the National Science Foundation) and the Swedish Government (evaluating the ARTES/PAMP national program for scientific research and graduate education in high-performance computing, as well as other programs

for the Swedish Research Council). I have also served as an external evaluator for faculty hiring at Uppsala University, Sweden and the University of Copenhagen, Denmark. I have given many invited presentations and lectures at a variety of international and domestic venues with details provided in my *curriculum vitae*.

15. I have served on the program committees and as program chair and track chair of many ACM, IEEE, and other conferences in scalable computing, including the International Symposium on Computer Architecture, Supercomputing/SC, SIGMETRICS Conference on Measurement and Modeling of Computer Systems, Principles and Practice of Parallel Programming, Symposium on Parallel Algorithms and Architectures, International Conference on Supercomputing, International Parallel Processing Symposium, International Conference on Parallel Processing etc., as well as several ACM and IEEE workshops on scalable computing, data analysis and mining, architecture, and languages and compilers.

16. In my education, research, and consulting work, I have used or designed many systems for parallel and clustered computation.

17. I am a named inventor of six issued patents: U.S. Patent No. 7,080,073; U.S. Patent No. 6,915,294; U.S. Patent No. 7,415,469; U.S. Patent No. 7,103,838; U.S. Patent No. 10,796,276; and U.S. Patent No. 10,796,277.

18. Based on my education and experience, I am an expert in the field of cluster computing. (*See below* ¶¶ 33-34.) I have used my education and experience working in the electrical engineering and computer science fields, and my understanding of the knowledge, creativity, and experience of a person having ordinary skill in the art in forming the opinions expressed in this declaration, as well as any other materials discussed herein.

III. OPINIONS AND ANALYSIS

19. I am informed by counsel that the parties in this litigation disagree on the proper interpretation of certain claim limitations in the '768 Patent. I have been asked to provide my opinion about how a person of ordinary skill in the art would interpret the disputed claim terms.

A. Legal Standards for Claim Construction

20. This section describes my understanding of relevant legal standards based on information provided to me by ACS's counsel—not my personal knowledge or expert opinion. I am not an attorney or a legal expert. My understanding of the relevant legal standards simply provides context for my technical opinions set forth herein.

1. General Standards

21. I understand that claim terms must be viewed from the perspective of a person of ordinary skill in the art to which the patent pertains, as of the patent's priority date.

22. I understand that claim terms are generally given their plain and ordinary meaning that a person of ordinary skill in the art would have ascribed to it when viewed in the context of the patent's claims, specification, and prosecution history. A construction different from the plain and ordinary meaning may be adopted based on the patentee's lexicography, or due to a disavowal of claim scope.

23. I understand that lexicography occurs when the patentee clearly expresses an intent to redefine a term or provides an express definition of a term. Disavowal occurs when clear and unequivocal statements in the specification or prosecution history of a patent indicate that the claimed invention requires the presence or absence of a particular feature. For example, disavowal may occur when the specification or prosecution history distinguishes prior art as lacking a particular feature. In this example, the disavowal would indicate that the invention claimed in the patent requires the feature used to distinguish the prior art.

24. I understand that a patent and its prosecution history (also known as the file history) are considered "intrinsic evidence" and are the most important sources for interpreting claim language in a patent. The prosecution history of related patents

and applications, including foreign patent applications, is also relevant. I also understand that any *inter partes* reviews regarding a patent are part of its prosecution history and intrinsic record.

25. I understand that sources extrinsic to a patent and its prosecution history (such as dictionary definitions, technical publications, and other unrelated patents) may also be used to help interpret the claim language, but that such extrinsic sources cannot be used to contradict the unambiguous meaning of the claim language that is evident from the intrinsic evidence.

26. I understand that when a person of ordinary skill in the art would be unable to ascertain the scope of a claim with reasonable certainty, the claim is indefinite and therefore invalid. I understand that a lack of reasonable certainty may arise when the prosecution history of a patent and the specification of a patent provide two conflicting meanings for a claim limitation.

27. I understand that the Court can correct a patent's claims during construction if there is an obvious error and correction is not subject to reasonable debate based on consideration of the claim language and the specification, and the prosecution history does not suggest a different interpretation of the claims.

2. Standards for Functional Claim Limitations

28. I understand that a patent claim may be subject to special legal rules when it uses "functional" language. Claim language is "functional" when it

describes a result to be achieved rather than a specific way of achieving that result. In the context of apparatus claims, functions (results to be achieved) are distinguished from structure (the machinery or mechanism used to achieve the results).

29. I understand that functional claim limitations are sometimes governed by 35 U.S.C. § 112, ¶ 6. Claim limitations governed by this statute are called “means-plus-function” limitations in apparatus claims. In contrast to the usual rules of claim construction, claim limitations that are subject to 35 U.S.C. § 112, ¶ 6 are not construed to have their plain and ordinary meaning.

30. I understand that not every recitation of function in a claim amounts to a means-plus-function limitation. A claim element that does not use the word “means” is presumed to be structural and not a means-plus-function term. I understand that this presumption may be rebutted if the claim term fails to recite sufficiently definite structure or recites function without reciting sufficient structure for performing that function. But a claim term is not a means-plus-function term if the words of the claim are understood by persons of ordinary skill in the art to have a sufficiently definite structure.

31. I understand that when claim language is treated as means-plus-function under 35 U.S.C. § 112, ¶ 6, the language is construed to cover only the structure described in the specification that correspond to the claimed function, and

equivalents thereof. Therefore, to construe a means-plus-function limitation, one must first identify the function recited in the claim. Next, one must identify the corresponding structure disclosed in the specification.

32. I understand that structure disclosed in the specification is “corresponding” structure only if the specification or prosecution history clearly links that structure to the function recited in the claim. The corresponding structure must include all clearly linked structure in the specification that performs the recited function, but nothing more.

B. Level of Ordinary Skill in the Art

33. I was asked by counsel to assess the field of the invention of the ’768 Patent and the level of ordinary skill in the art. In making this assessment, I have considered the ’768 Patent, my experience, and the other materials referenced below. (*See below* ¶ 193.)

34. In my opinion, the field of the invention of the ’768 Patent is “cluster computing.” This is supported by the “Field of Disclosure” section of the patent, the specification generally, and the claims. The “Field of Disclosure” section states: “The present disclosure relates to the field of cluster computer generally” The specification of the ’768 Patent consistently discloses aspects of a computer cluster or methods of operating a computer cluster. Every independent claim of the ’768 Patent expressly recites a “computer cluster” or a “computer cluster node.”

35. I understand that the '768 Patent claims priority to a provisional patent application filed on June 13, 2006. I also understand that the '768 Patent also cites to a provisional patent application filed on October 11, 2006 as a related U.S. patent application. If this latter date is viewed as the priority date of this patent instead of the former, my opinions and analysis herein would be unchanged by this four month difference.

36. In my opinion, as of June 13, 2006, a person of ordinary skill in the art would have had a Bachelor's degree in computer science, electrical engineering, or an equivalent field, and two years of academic or industry experience in cluster computing. In conducting my analysis set forth in this declaration, I applied my identification of the level of ordinary skill in the art.

37. At the time of the invention and presently I did and do have more training, expertise, and knowledge than a person of ordinary skill in the art. However, I understand what a person of ordinary skill in the art would have known and understood at the relevant time and my testimony here is from the perspective of a person of ordinary skill in the art at the relevant time.

38. I understand that NVIDIA Corp. previously filed two petitions for *inter partes* review directed at the '768 Patent and both were denied institution. NVIDIA Corp. submitted two declarations from its expert, Dr. Henry Tufo, advancing a slightly different level of ordinary skill in the art. I disagree with Dr. Tufo's

proposed level of ordinary skill in the art in part because he incorrectly defined the relevant field of the invention as “parallel and/or distributed computing.” (IPR2021-00019 Tufo Declaration at 23; IPR2021-00020 Tufo Declaration at 24.) The specification of the ’768 Patent makes clear that cluster computing, distributed computing, and parallel computing are different fields.

39. Regardless, my opinions and conclusions below would remain the same even if the priority date, field of invention, or level of ordinary skill were slightly different.

C. Overview of the Technology

40. The ’768 Patent describes and claims a fundamental improvement over previous efforts to allow multiple nodes to work together to perform computations in parallel. Specifically, the inventors designed a mechanism to allow the nodes of a computer cluster to communicate tasks and data with one another in a peer-to-peer architecture. According to the background section of the patent, one previous attempt to allow multiple nodes to work together to perform computations was “a form of grid computing known as ‘distributed computing.’” (’768 Patent at col. 1, ll. 44-62.) The form of grid computing described in the patent relied on a “master node that manages a plurality of slave nodes or computational nodes.” (*Id.* at col. 1, ll. 51-53.) The “master kernel . . . handles all input, output, and scheduling of the other kernels (the computational kernels or slave kernels). Computational kernels

receive commands and data only from the node running the master kernel.” (*Id.* at col. 1, ll. 55-59.) The nodes “generally do not communicate with one another as peers.” (*Id.* at col. 1, ll. 46-47.) The patent distinguishes this form of grid computing from a “computer cluster having peer-to-peer node architecture.” (*Id.* at col. 12, ll. 33-40.)

41. The ’768 Patent expressly claims the “peer-to-peer node architecture” disclosed in the specification. Specifically, Claim 1 recites a computer cluster with multiple nodes and “a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture.” Claim 1 further recites that “one or more of the nodes are configured to . . . after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other.”

42. The specification discloses a mechanism for the nodes to communicate using a “peer-to-peer architecture” and in some embodiments using “asynchronous calls.” In one embodiment multiple “cluster node modules” establish connections from each cluster node module to every other cluster node module and exchange messages in a process that “provides the peer-to-peer behavior of the cluster node modules.” (*Id.* at col. 23, ll. 51-52, col. 24, ll. 39-40, 52-53, col. 25, ll. 1-2, 9-10, 23-24, 37-38.) Multiple claims recite the “cluster node modules.”

43. A person of ordinary skill in the art would understand, in view of the specification, that “peer-to-peer behavior” is an essential feature of both the “peer-to-peer architecture” limitation of Claims 1 and 35 and the narrower “cluster node modules” limitations. “Peer-to-peer behavior” includes at least that each node (through, *e.g.*, its cluster node module) can communicate tasks and data with other nodes (through, *e.g.*, their cluster node modules) without the tasks and data being required to go through a central server or master node.

D. Claims

44. The ’768 Patent contains 39 claims. Claims 1, 26, 29, and 35 are independent claims and each recite various aspects of a “computer cluster.” Claims 1, 26, and 29 are directed to the “computer cluster” itself. Claim 35 is directed to an individual “computer cluster node.”

E. Prosecution History

45. During the prosecution of the ’768 Patent, the applicant amended the claims to distinguish certain prior art, to address a rejection for lack of patent-eligible subject matter, and to address a rejection for indefiniteness. The Examiner also proposed an amendment to place the claims in condition for allowance, which ACS accepted. (November 29, 2018 Notice of Allowability at 11-12; November 29, 2018 Examiner Initiated Interview Summary.) The amendments are described below in connection with the relevant claim limitations. (*See below* ¶¶ 180-192.)

46. The '768 Patent was the subject of two petitions for *inter partes* review by NVIDIA Corp., addressing Claims 1, 4-10, 18-22, 24-25, 30-31, and 33-34 (IPR2021-00019 Petition for *Inter Partes* Review at 10), and addressing Claims 26-27, 29, 35-37, and 39 (IPR2021-00020 Petition for *Inter Partes* Review at 12-13). Both of these petitions for *inter partes* review were denied. (IPR2021-00019 and IPR2021-00021 Decisions Denying Institution of *Inter Partes* Review.) The relevant portions of the prosecution history are described below in connection with the relevant claim limitations.

F. Claim Construction

1. Peer-To-Peer Architecture

47. Claims 1 and 35 recite a “peer-to-peer architecture.” I understand the parties proposed the following constructions for this claim term.

ACS Construction	NVIDIA Construction
architecture in which each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node	an architecture in which each node is configured to communicate with other nodes

48. The parties disagree about two requirements of the construction of “peer-to-peer architecture”: what type of information is communicated between nodes, and whether communication between nodes is without the tasks and data being required to go through a central server or master node.

49. In the field of cluster computing, a person of ordinary skill in the art would have understood the adjective “peer-to-peer” to mean that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node. A person of ordinary skill in the art would have looked to the ’768 Patent’s specification to confirm this understanding.

50. As explained below, a person of ordinary skill in the art would have concluded from the specification that communicating tasks and data with other nodes without the tasks and data being required to go through a central server or master node is a primary function of the “peer-to-peer architecture” of the ’768 Patent. Therefore, “peer-to-peer architecture” means “an architecture in which each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.”

51. A person of ordinary skill in the art would have understood that the “peer-to-peer architecture” required by Claims 1 and 35 is one feature of the interconnected “cluster node modules” disclosed by the ’768 Patent and recited by Claim 4.¹ FIG. 2 illustrates that each cluster node module is connected to every other cluster node module, as shown by the red annotations in the figure below.

¹ Claim 1 is broader than Claim 4 because, while it requires a “peer-to-peer architecture,” it does not require that architecture to be implemented using the “cluster node modules” recited by Claim 4.

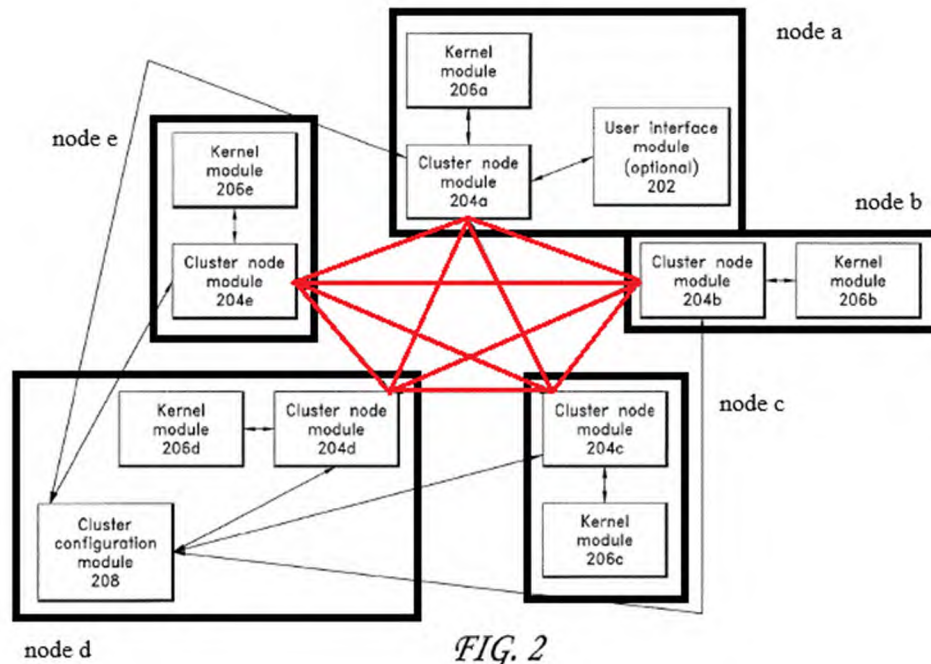


FIG. 2

(’768 Patent, FIG. 2 (annotations added).) The specification confirms that “each cluster node module 204a-e is connected to all other cluster node modules” and “[t]he cluster node modules 204a-e establish communication with one another” through “direct connections” between each cluster node module. (*Id.* at col. 23, ll. 13-14, 51-56.)

52. As illustrated and described, the cluster node modules provide a direct connection from each node to every other node. In addition, the specification discloses that a process for passing messages among the cluster node modules “provides the peer-to-peer behavior of the cluster node modules,” allowing them to “interact on a pair-wise or collective basis.” (*Id.* at col. 25, ll. 22-41.)

53. In view of the specification, a person of ordinary skill in the art would have understood that the message-passing process allows each cluster node module to communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node. Specifically, the specification discloses that “[c]ommunications can occur between any two or more cluster node modules” and that “[e]ach of the cluster node modules 204a-e is in communication with respective kernel modules 206a-e,” thereby enabling “MPI calls and advanced cluster commands” to be “used to parallelize program code received from an optional user interface module 208 and distribute tasks among the kernel modules 206a-e.” (*Id.* at col. 6, ll. 9-25.)

54. In addition, the specification distinguishes a peer-to-peer architecture from the master-slave architecture in the prior art typically used for “a form of grid computing known as ‘distributed computing.’” (*Id.* at col. 1, ll. 44-62; *see also id.* at col. 12, ll. 30-33 (gridMathematica connects kernels “in a master-slave relationship rather than a peer-to-peer relationship”).) The specification describes the following characteristics of the prior art master-slave architecture of grid computers:

Grid computers include at least one node known as a master node that manages a plurality of slave nodes or computational nodes. In gridMathematica, each of a plurality of kernels runs on a single node. ***One kernel is designated the master kernel, which handles all input, output, and scheduling of the other kernels (the computational***

kernels or slave kernels). Computational kernels receive commands and data only from the node running the master kernel.

(*Id.* at col. 1, ll. 51-59 (emphasis added).) By distinguishing the peer-to-peer architecture of the invention from the prior art master-slave architecture, the specification indicates that the peer-to-peer architecture is different than the identified characteristics of the master-slave architecture. Accordingly, by contrast to the master-slave architecture, each node in the disclosed peer-to-peer architecture is able to handle “scheduling” and communicating (including distributing, sending, and receiving) “commands and data” to and from other nodes without requiring them to go through a central server or master node. A person of ordinary skill in the art would have understood that the communication of tasks and data falls within these “scheduling” and communicating “commands and data” functions.

55. Accordingly, a person of ordinary skill in the art would have understood that the communication of tasks and data is a primary function that the nodes of the disclosed “peer-to-peer architecture” of the ’768 Patent can handle without requiring a central server or master node. The term “peer-to-peer architecture,” should therefore be construed as “an architecture in which each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.”

56. Defendants’ proposed construction of “an architecture in which each node is configured to communicate with other nodes” is inconsistent with the plain

and ordinary meaning of peer-to-peer in the context of cluster computing, as well as with the patent specification. A person of ordinary skill in the art would have understood that a fundamental feature of a peer-to-peer architecture in cluster computing is the ability for the nodes to communicate with each other and without the involvement of a central server or master node. This is what distinguishes a peer-to-peer architecture from other multi-node architectures, such as a prior art master-slave architecture, where communication between the nodes is governed by a central server or master node. The specification uses the term consistently with its plain and ordinary meaning. As discussed above, communicating tasks and data with other nodes without the tasks and data being required to go through a central server or master node is a primary function of the “peer-to-peer architecture” disclosed in the ’768 Patent. (*See above* ¶¶ 50-55.) Defendants’ proposed construction is incorrect because it would encompass architectures where the nodes communicate indirectly, through an intermediary like a central server or master node. These prior art communication methods are contrasted from the peer-to-peer architecture of the invention.

57. Construing “peer-to-peer architecture” as “an architecture in which each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node” is consistent with the prosecution history. With respect to the ’768 Patent’s two *inter partes* reviews,

ACS proposed a construction for “peer-to-peer architecture” substantially similar to the one proposed herein. (IPR2021-00019 Patent Owner Preliminary Response at 8-12; IPR2021-00020 Patent Owner Preliminary Response at 8-12.) NVIDIA Corp. neither proposed a construction for “peer-to-peer architecture” nor objected to ACS’s construction in its Reply. (See IPR2021-00019 Petition for *Inter Partes* Review at 14; IPR2021-00020 Petition for *Inter Partes* Review at 16-17; see IPR2021-00019 and IPR2021-00020 Petitioner’s Reply.) The Patent Trial and Appeal Board found that it was not necessary to its decision to construe “peer-to-peer architecture.” (IPR2021-00019 Decision Denying Institution of *Inter Partes* Review at 11; IPR2021-00020 Decision Denying Institution of *Inter Partes* Review at 12.)

58. Accordingly, the intrinsic evidence supports my opinion that a person of ordinary skill in the art would have understood “peer-to-peer architecture” to mean “architecture in which each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.”

59. I also understand that Defendants have identified various pieces of extrinsic evidence to support their construction of “peer-to-peer architecture.” I have reviewed Defendants’ extrinsic evidence and determined that it does not contradict ACS’s proposed construction, and in some instances supports ACS’s construction.

60. Defendants cite *IEEE 100: The Authoritative Dictionary of IEEE*

Standards Terms definition of peer-to-peer communication:

Peer-to-peer communication (A) Communication between two or more processes or programs by which both computers can exchange data freely. *Note:* Any physical differences between the computers are rendered transparent to the application.

(B) Communication between two or more network nodes in which either node can initiate sessions, and is able to poll or answer to polls.

(*IEEE 100: The Authoritative Dictionary of IEEE Standards Terms*, peer-to-peer communication.)

61. Both definitions are consistent with ACS's proposed construction. Definition (A) suggests that peer-to-peer communications are inconsistent with a master-slave architecture by stating that "both computers can exchange data freely." (*Id.* (emphasis added).) Definition (B) states that "either node can initiate sessions, and is able to poll or answer to polls," indicating a symmetry among nodes that is not consistent with a master-slave architecture. (*Id.*) Definition (B) also indicates that peer-to-peer communications include communication of both tasks and data. Polling is a request for a status, *i.e.*, it communicates a task to be performed by a node. A response to a poll is a status signal, *i.e.*, data. Defendants' proposed construction omits these important features of the IEEE dictionary definition. (*See id.*)

62. Defendants also cite to journal articles that pertain to peer-to-peer file sharing. (*See* Dorrigiv et al., *Search Algorithms for Unstructured Peer-to-Peer*

Networks; Tewari and Kleinrock, *Optimal Search Performance in Unstructured Peer-to-Peer Networks With Clustered Demands*; Van Roy, *Overcoming Software Fragility with Interacting Feedback Loops and Reversible Phase Transitions*; Gkantsidis et al., *Hybrid Search Schemes for Unstructured Peer-to-Peer Networks*.) Peer-to-peer file sharing does not involve computation as it is used in the '768 Patent, for example the evaluation of mathematical expressions. The field of the invention of the '768 Patent is "cluster computing," which involves computing in parallel. NVIDIA's reliance on peer-to-peer file sharing articles is therefore misplaced because it does not provide evidence of what a person of ordinary skill in the art would have understood "peer-to-peer architecture" to mean in the context of the '768 Patent.

63. Based upon the above-described portions of the specification, the Court should construe "peer-to-peer architecture" to mean "architecture in which each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node."

2. Cluster Node Module

64. Claims 4, 7-8, 10, 27, and 30-31 recite "cluster node module."² I understand the parties proposed the following constructions for this claim term.

² Claims 6, 14-17, 28, and 32 also recite "cluster node module," but I understand that these claims are not asserted at this time.

ACS Construction	NVIDIA Construction
a module that establishes intercommunication among nodes in a computer cluster and allows exchanging messages among nodes using a peer-to-peer architecture	a module running on each cluster node, configured to communicate messages with the single-node kernel on the same node, and with other cluster node modules

65. The claim term “cluster node module” is used several times through the ’768 Patent:

Claim	Language ³
4	“The computer cluster of claim 1, wherein each of the nodes comprises one or more <u>cluster node modules</u> .”
7	“The computer cluster of claim 6, wherein the plurality of <u>cluster node modules</u> act as a cluster”
8	“The computer cluster of claim 6, wherein the plurality of <u>cluster node modules</u> communicate with one another to act as a cluster”
10	“The computer cluster of claim 9, wherein each <u>cluster node module</u> accepts instructions from the user interface and interprets one or more of the instructions”
27	“wherein the single-node kernel is configured to send the first packet to a local <u>cluster node module</u> ; and wherein the local <u>cluster node module</u> is configured to forward the expression to the target node”
30	“The computer cluster of claim 4, wherein each of the plurality of nodes implements asynchronous calls that enable the single-node kernel to perform computation tasks while the <u>cluster node modules</u> are simultaneously communicating with one another”

³ Emphasis added to highlight the use of the claim term “cluster node module.”

31	“The computer cluster of claim 4, wherein intercommunication among the plurality of single-node kernels during thread execution is enabled by the plurality of <u><i>cluster node modules</i></u> ”
----	---

66. A person of ordinary skill in the art in the field of cluster computing would not have understood “cluster node module” to have any ordinary meaning. The phrase “cluster node module” was not a common or well-known technical phrase having any specific meaning in the relevant art at the time of the invention. Indeed, even today, more than 14 years after the earliest effective filing date of the ’768 Patent, a search for patents or patent application publications that use the phrase “cluster node module” returns only patents and applications filed by the inventors of the ’768 Patent. Accordingly, there was no ordinary meaning of “cluster node module” in the relevant field at the relevant time. Further, while the individual terms “cluster,” “node,” and “module,” were known in the relevant field, a person of ordinary skill in the art would have understood that no ordinary meaning for the combined phrase “cluster node module” could reliably be composed by attempting to combine the meanings of the individual terms.

67. In view of the claim language and specification, a person of ordinary skill in the art would have understood that the inventors coined the phrase “cluster node module” to encapsulate essential features of the modules that interconnect the nodes in a preferred embodiment of the invention. I understand that, because “cluster node module” is a coined phrase, reliance on the specification is necessary

to ascertain its meaning. As explained below, a person of ordinary skill in the art would have understood “cluster node module” to mean “a module that establishes intercommunication among nodes in a computer cluster and allows exchanging messages among nodes so each node can communicate tasks and data with other nodes using a peer-to-peer architecture.”

68. The specification discloses several *optional* components of the cluster node modules. For example:

FIG. 3 shows one embodiment of a cluster node module 204 implementing MPI calls and advanced MPI functions. ***In the embodiment shown in FIG. 3***, cluster node module 204 includes MPI module 302, advanced functions module 304, received message queue 306, and message receiving queue 308.

(’768 Patent at col. 12, ll. 41-46 (emphases added).) Because the components depicted in FIG. 3 are part of “one embodiment,” a person of ordinary skill in the art would have understood that they are optional components of the cluster node modules. Dependent claims of U.S. Patent No. 8,082,289 (which is related to the ’768 Patent) specifically reciting that the cluster node modules comprise a MPI module, advanced functions module, received message queue, and message receiving queue also demonstrate that these components are optional parts of the cluster node modules. (U.S. Patent No. 8,082,289 at cl. 8 (received message queue), cl. 9 (message receiving queue), cl. 13 (advanced functions module), cl. 26 (MPI module).) Accordingly, a person of ordinary skill in the art would have understood

that a “cluster node module” does not require a MPI module, advanced functions module, received message queue, or message receiving queue.

69. Rather than interpreting “cluster node module” to include optional components such as those depicted by FIG. 3, a person of ordinary skill in the art would look to the specification to understand the essential features of a “cluster node module.” The specification unambiguously discloses that cluster node modules are configured to establish intercommunication with one another, communicate messages among themselves, and, through such message-passing, “provide[] the peer-to-peer behavior of the cluster node modules.” (’768 Patent at col. 23, ll. 51-52, col. 24, ll. 39-40, 52-53, col. 25, ll. 1-2, 9-10, 23-24, 37-38.) Significantly, the specification does not say these disclosed features apply just to “one embodiment” of the cluster node modules or otherwise suggest they are optional. Therefore, a person of ordinary skill in the art would have understood that the capability to establish intercommunication among all cluster node modules and to exchange messages to provide “peer-to-peer behavior” are essential features of the cluster node modules.

70. A person of ordinary skill in the art would have further examined both the ordinary meaning of “peer-to-peer” and the specification to determine what is meant by the cluster node modules’ “peer-to-peer behavior.” As explained above with respect to the “peer-to-peer architecture” limitation, a person of ordinary skill

in the art would have concluded that “peer-to-peer behavior” includes that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node. (*See above* ¶¶ 47-63.)

71. Accordingly, the Court should construe “cluster node module” to mean “a module that establishes intercommunication among nodes in a computer cluster and allows exchanging messages among nodes using a peer-to-peer architecture.”⁴

72. Defendants’ proposed construction of “cluster node module” as “a module running on each cluster node, configured to communicate messages with the single-node kernel on the same node, and with other cluster node modules” is inconsistent with the specification. A person of ordinary skill in the art would have understood that a fundamental feature of the “cluster node module” is to provide peer-to-peer behavior. Although Defendants’ proposed construction includes exchanging messages between the modules, it is equally compatible with exchanging messages in a prior art master-slave architecture. Defendants’ proposed construction is therefore incorrect because it would encompass architectures where the nodes must communicate in a master-slave architecture.

73. Construing “cluster node module” as “a module that establishes

⁴ A person of ordinary skill in the art would recognize there is some overlap between the “cluster node modules” limitation of Claim 4 and the “peer-to-peer architecture” limitation of Claims 1 and 35, but the “cluster node modules” limitation is narrower.

intercommunication among nodes in a computer cluster and allows exchanging messages among nodes using a peer-to-peer architecture” is also consistent with the prosecution history.

74. With respect to the ’768 Patent’s two *inter partes* reviews, ACS proposed a construction for “cluster node module” substantially similar to the one proposed herein.⁵ (IPR2021-00019 Patent Owner Preliminary Response at 16; IPR2021-00020 Patent Owner Preliminary Response at 15.) NVIDIA Corp. neither proposed a construction for “cluster node module” nor objected to ACS’s construction in its Reply. (IPR2021-00019 Petition for *Inter Partes* Review at 14; IPR2021-00020 Petition for *Inter Partes* Review at 16-17; *see* IPR2021-00019 and IPR2021-00020 Petitioner’s Reply.) The Patent Trial and Appeal Board found that it was not necessary to its decision to construe “cluster node module.” (IPR2021-00019 Decision Denying Institution of *Inter Partes* Review at 11; IPR2021-00020 Decision Denying Institution of *Inter Partes* Review at 12.)

⁵ ACS’s proposed construction of “cluster node module” differs slightly from the one ACS proposed during the two *inter partes* reviews of the ’768 Patent: “a module that cooperates with other cluster node modules to establish intercommunication among nodes in a computer cluster and to exchange messages such that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.” The aspect of cooperation with other cluster node modules was removed and the construction was shortened by including “using a peer-to-peer architecture” construed as by ACS herein. I understand that these changes were made to simplify the construction. I do not view these changes as substantive.

75. Accordingly, the intrinsic evidence supports my opinion that a person of ordinary skill in the art would have understood “cluster node module” to mean “a module that establishes intercommunication among nodes in a computer cluster and allows exchanging messages among nodes using a peer-to-peer architecture.”

76. Based upon the above-described portions of the specification and file history, the Court should construe “cluster node module” to mean “a module that establishes intercommunication among nodes in a computer cluster and allows exchanging messages among nodes using a peer-to-peer architecture.”

3. Kernel

77. Claims 1, 26, 29, 35, and some various dependent claims recite “kernel.” I understand the parties proposed the following constructions for this claim term.

ACS Construction	NVIDIA Construction
Plain and ordinary meaning	program code for interpreting high-level code, commands, and/or instructions supplied by a user or a script into low-level code, such as, for example, machine language or assembly language

78. The term “kernel” was a common, well-known term to a person of ordinary skill in the relevant art at the time of the invention. In the field of cluster computing, a person of ordinary skill in the art would have understood “kernel” as general term used to refer to program code, and that the term could be used in

different contexts to describe different “types” of kernels. A kernel may do many things: interpret, calculate, implement algorithms, manage input-output, etc. A person of ordinary skill in the art would therefore have read the term “kernel” in the context of the patent’s claims, specification, and prosecution history.

79. The claims of the ’768 Patent describe the kernel in detail and provide context for the use of the term in the ’768 Patent. There are many references to “kernel”, and I have done my best to categorize them here.

- The claims refer to a “kernel” as program code stored on a computer readable medium. (*E.g.*, ’768 Patent at cl. 1 (“non-transitory computer-readable medium comprising program code for a single-node kernel”), cl. 26 (same), cl. 29 (same), cl. 35 (“one or more non-transitory memory devices comprising program code for a single-node kernel”), cl. 5 (“each single-node kernel is stored in a non-transitory computer-readable medium”), cl. 12 (“computer-readable medium comprising program code for the single-node kernel”), cl. 16 (“non-transitory computer-readable medium comprising program code for the single-node kernel”), cl. 1 (“a first memory comprising program code . . . for a first single-node kernel”), cl. 26 (same), cl. 29 (same), cl. 35 (“a second memory comprising program code . . . for a second single-node kernel”).)

- The claims refer to the “kernel” as causing the hardware processor to evaluate mathematical expressions when executed. (*E.g.*, *id.* at cl. 1 (“single-node kernel that, when executed, is capable of causing the hardware processor to evaluate mathematical expressions”), cl. 26 (“single-node kernel that, when executed, is capable of causing a hardware processor to evaluate mathematical expressions”).)
- The claims refer to the “kernel” as configured to interpret user instructions and distribute calls. (*E.g.*, *id.* at cl. 1 (“the first single-node kernel configured to interpret user instructions and distribute calls”), cl. 26 (same), cl. 29 (same), cl. 35 (“the second single-node kernel configured to interpret user instructions and distribute calls”).)
- The claims refer to the “kernel” as something to which user instructions are communicated. (*E.g.*, *id.* at cl. 1 (“communicate at least some of the user instructions to one or more single-node kernels”), cl. 19 (same), cl. 23 (same), cl. 24 (same), cl. 26 (same), cl. 29 (same), cl. 35 (“communicate at least some of the user instructions to the single-node kernel”).)
- The claims refer to the “kernel” as configured to accept and execute a request. (*E.g.*, *id.* at cl. 5 (“each single-node kernel is . . . configured to accept and execute a request.”).)

- The claims refer to the “kernel” as executing or calling commands. (*E.g.*, *id.* at cl. 25 (“commands for execution by a plurality of single-node kernels”), cl. 27 (“the first command is configured to be called from within a single-node kernel”), cl. 28 (“wherein the second command is configured to be called from within a single-node kernel”) cl. 32 (“single-node kernels are configured to call a send command”), cl. 32 (“single-node kernels is configured to call a receipt command . . . and to then wait for a reply expression to evaluate”).)
- The claims refer to the “kernel” as configured to send packets to a cluster node module. (*E.g.*, *id.* at cl. 27 (“single-node kernel is configured to send the first packet to a local cluster node module”), cl. 28 (“wherein the single-node kernel is configured to send the second packet to a local cluster node module”).)
- The claims refer to the “kernel” as performing computation tasks. (*E.g.*, *id.* at cl. 30 (“the single-node kernel to perform computation tasks”).)
- The claims refer to the “kernel” as configured to receive the completed expression and to update variables used by the single-node kernel during evaluation of expressions. (*E.g.*, *id.* at cl. 32 (“single-node kernels is configured to receive the completed expression and to update variables used by the single-node kernel during evaluation of

expressions”).)

- The claims refer to “kernel” as being associated with or local to a cluster node module. (*E.g.*, *id.* at cl. 32 (“local cluster node module associated with the single-node kernel”).)
- The claims refer to “kernel” as comprising a specific software program. (*E.g.*, *id.* at cl. 2 (“the single-node kernel comprises a Mathematica kernel”).)

80. These numerous and varied references to a “kernel” in the claims are consistent with my opinion that a person of ordinary skill in the art would have understood “kernel” to have its plain and ordinary meaning. The detailed claim language illustrates that any specific kernel functions or configurations are claimed explicitly. For example, Claim 1 recites a kernel “that, when executed, is capable of causing the hardware processor to evaluate mathematical expressions,” and is “configured to interpret user instructions and distribute calls.” Considering the detailed claim language, a person of ordinary skill in the art would have understood that the claims use the term “kernel” according to its plain and ordinary meaning.

81. Defendants’ proposed construction adopts an unnecessarily narrow definition of the term “kernel.” Defendants propose “program code for interpreting high-level code, commands, and/or instructions supplied by a user or a script into low-level code, such as, for example, machine language or assembly language.”

This describes one, very specific type of kernel, but is not what a person of ordinary skill in the art would have understood to be the plain and ordinary meaning of the term “kernel” generally, or in view of the intrinsic record.

82. Defendants’ proposed construction is inconsistent with the claim language because it incorporates several terms that are already explicitly in the claims. This leads to awkward and redundant claim language. For example, with Defendants’ proposed construction, a limitation from Claim 1 would read:

a first memory comprising program code for a user interface and program code for a first single-node program code for interpreting high-level code, commands, and/or instructions supplied by a user or a script into low-level code, such as, for example, machine language or assembly language, the first single-node program code for interpreting high-level code, commands, and/or instructions supplied by a user or a script into low-level code, such as, for example, machine language or assembly language configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution.

(text from Defendants’ proposed construction underlined and in red.) Independent Claims 26, 29, and 35 would read similarly.

83. A person of ordinary skill in the art would have read the specification as using the term kernel in a manner that is consistent with its plain and ordinary meaning. (*E.g., id.*, Abstract (“a kernel for interpreting program code instructions.”), col. 2, ll. 34-37 (“A first kernel resides in the at least one computer-readable medium and is configured to translate commands into code for execution on the first processor. . . . A second kernel resides in the at least one computer-readable medium.

The second kernel is configured to translate commands into code for execution on the second processor. . . . A third kernel resides in the at least one computer-readable medium. The third kernel is configured to translate commands into code for execution on the third processor.”), col. 3, ll. 16-17 (“Each node is configured to access a computer-readable medium comprising . . . program code for a single-node kernel module configured to interpret user instructions.”), col. 4, ll. 56-58 (“In one embodiment, one or more software modules such as kernels, run on nodes within the interconnected computer systems.”).)

84. One portion of the specification describes a “Kernel Module 206.” The specification states:

A kernel module 206 typically includes program code for interpreting high-level code, commands, and/or instructions supplied by a user or a script into low-level code, such as, for example, machine language or assembly language. In one embodiment, each cluster node module 204a-e is connected to all other cluster node modules, while each kernel module 206a-e is allocated and connected only to one cluster node module 204. In one embodiment, there is one cluster node module-kernel module pair per processor. For example, in an embodiment of a computer cluster 100 including single-processor computer systems, each cluster node module-kernel module pair could reside on a single-processor computer. If a computer contains multiple processors or processing cores, it may contain multiple cluster node module-kernel module pairs, but the pairs can still communicate over the cluster node module's network connections.

(*Id.* at col. 23, ll. 8-26.)

85. Defendants appear to base their proposed construction – “program code for interpreting high-level code, commands, and/or instructions supplied by a user

or a script into low-level code, such as, for example, machine language or assembly language” – on the description of the “kernel module 206.” However, a person of ordinary skill in the art would have understood the description of kernel module 206 to describe one thing a kernel might do, but a person of ordinary skill in the art would have understood a kernel to be more general than this. A person of ordinary skill in the art would not have understood the description of kernel module 206 to redefine the plain and ordinary meaning of “kernel.”

86. Another portion of the specification associates an interpreter with a “kernel.” The specification discloses:

Some application programs include an interpreter that executes instructions provided to the program by a user, a script, or another source. Such an interpreter is sometimes called a “kernel” because, for example, the interpreter can manage at least some hardware resources of a computer system and/or can manage communications between those resources and software (for example, the provided instructions, which can include a high-level programming language).

(*Id.* at col. 1, ll. 29-37.)

87. A person of ordinary skill in the art would have understood this to disclose that an interpreter is sometime called a kernel. As with kernel module 206, this is merely exemplary. A person of ordinary skill in the art would not have understood this to mean that a “kernel” is necessarily an interpreter, or to redefine the plain and ordinary meaning of “kernel” in any manner.

88. The specification also identifies Mathematica as a specific kernel

embodiment. (*E.g., id.* at col. 3, ll. 1-2 (“In one embodiment, the single-node kernel includes a Mathematical kernel.”) Again, this is merely exemplary, and the invention is not limited to a Mathematica kernel. (*See id.* at col. 4, ll. 14-18 (“For purposes of illustration, some embodiments are described herein in the context of cluster computing with Mathematica software. The present disclosure is not limited to a single software program . . .”).) A person of ordinary skill in the art would not have understood this to mean that a “kernel” is necessarily Mathematica, or to redefine the plain and ordinary meaning of “kernel” in any manner.

89. Nothing in the ’768 Patent’s prosecution history suggests that “kernel” should be understood as anything other than its plain and ordinary meaning.

90. I also understand that the Defendants have identified various pieces of extrinsic evidence to support their construction of “kernel.” I have reviewed Defendants’ extrinsic evidence and determined that it does not contradict ACS’s proposed construction of plain and ordinary meaning for “kernel.”

91. Defendants’ reliance on extrinsic evidence related to Mathematica is misplaced. Defendants cite sections of Mathematica 5’s Reference Guide, which states “*Mathematica* is fundamentally an interpreter, which scans through expressions” and “*Mathematica* is a modular software system in which the kernel which actually performs computations is separate from the front end which handles interaction with the user.” (Mathematica 5 Reference Guide Section A.9.3; The

Mathematica Book Section 1.31.)

92. Defendants also cite two patents to Theodore Gray U.S. Patents No. 9,141,355 and 7,747,981. Theodore Gray is affiliated with Wolfram Research, the company that makes Mathematica, and these patents are assigned to Wolfram Research. Both of these patents relate to Mathematica. (U.S. Patent No. 9,141,355 at col. 3, ll. 34-38; U.S. Patent No. 7,747,981 at col. 1, ll. 14-16 (“This disclosure will refer to a particular implementation of these techniques in the MATHEMATICA® software system available from Wolfram Research Inc.”).)

Defendants cite U.S. Patent No. 9,141,355 for the statement that:

Returning to FIG. 1, the kernel 120 receives the input from the front end 110. Some of the input or expression may include functions that are to be performed on objects associated with the function. The kernel 120 processes the input in order to parse the input into binary representations of the input and may form data structures within the system. More particularly, the kernel 120 parses both the functions and the associated objects into their binary representations. The kernel 120 then performs computations with the data structures, such as by applying algorithms corresponding to designated functions on the associated objects.

(U.S. Patent No. 9,141,355 at col. 5, ll. 27-37.) Defendants cite U.S. Patent No. 7,747,981 for the statement that:

MATHEMATICA® is a powerful computational tool that can evaluate general symbolic expressions, as well as mathematical and numeric expressions. A unifying feature of MATHEMATICA® is that everything is internally represented as a symbolic expression, with all more specific data types treated as special cases—symbols to which additional rules apply. MATHEMATICA® is an interpreted language, with a notion of “evaluation” of symbolic expressions. Evaluation

consists in applying to any symbolic expression all transformation rules that fit that expression.

(U.S. Patent No. 7,747,981 at col. 1, ll. 21-30.) These statements do not alter the ordinary meaning of the term “kernel” as used in the ’768 Patent.

93. First, these statements are made by Wolfram Research, not ACS, to describe Mathematica, not the “kernel” in the ’768 Patent.

94. Second, the ’768 Patent claims make clear that the inclusion of a Mathematica kernel is a specific embodiment of “kernel” generally. (’768 Patent at cl. 2 (“The computer cluster of claim 1, wherein the single-node kernel comprises a Mathematica kernel.”).) A person of ordinary skill in the art would have read this narrower dependent claim to imply that the broader independent claim’s “kernel” included more than just Mathematica kernels.

95. Third, the ’768 Patent’s specification makes clear that the use of a Mathematica kernel is only one embodiment of the invention. (*See id.* at col. 2, ll. 18-20 (“One embodiment adapts a software module designed to run on a single node, such as, for example, the Mathematica kernel, to support cluster computing”), col. 4, ll. 14-18 (“For purposes of illustration, some embodiments are described herein in the context of cluster computing with Mathematica software. The present disclosure is not limited to a single software program; the systems and methods can be used with other application software”).)

96. Fourth, a person of ordinary skill in the art would have found some of

the statements inconsistent with Defendants’ proposed construction. One cited passage from the reference guide refers to the Mathematica kernel only as performing computations. (The Mathematica Book Section 1.31.) U.S. Patents No. 9,141,355 and 7,747,981 refer to Mathematica as performing both parsing and computations, although U.S. Patent 7,747,981 does not specify what component within Mathematica, *i.e.*, the computational tool or the language, performs the stated functions. (U.S. Patent No. 9,141,355 at col. 5, ll. 27-37; U.S. Patents No. 7,747,981 at col. 1, ll. 21-30.) These descriptions of Mathematica are inconsistent with the Defendants’ proposed construction of “kernel,” which eschews computation.

97. Defendants’ reliance on extrinsic evidence related to Maple is also misplaced. Defendants cite to the Maple V Learning Guide:

The kernel is the base of Maple’s system. It contains fundamental and primitive commands: the Maple language interpreter (which converts the commands you type into machine instructions your computer processor can understand), algorithms for numerical calculation, and routines to display results and perform other input and output operations

(Maple V Learning Guide at 88.) Like Mathematica, Maple is a third-party computational program. This citation suffers from many of the same defects as Defendants’ citations to the Mathematica Reference Guide and patents. This is a third-party’s description of a separate prior art software program. This disclosure recognizes that the Maple interpreter is part of the kernel, not the kernel itself. Moreover, this disclosure even more clearly recognizes that the kernel performs both

interpretive functions and algorithmic functions for calculations, as well as input-output functions. This is inconsistent with Defendants' proposed construction, which conflates the interpreter, the first function, with the "kernel," to the exclusion of algorithmic and input-output functions.

98. Defendants' reliance on extrinsic evidence related to interpreters is misplaced. Defendants cite *The American Heritage College Dictionary's* definition of "interpreter": "*Comp. Sci.* A program that translates an instruction into a machine language and executes it before the next instruction." (*The American Heritage College Dictionary*, interpreter.) First, this is not a definition of "kernel," this is a definition for an "interpreter." Second, although this purports to be a computer science definition, a person of ordinary skill in the art would not have placed much weight on a general-purpose dictionary definition from 1993 to define a term in the cluster computing context, a highly specialized form of computer science and computer system architecture, as it would have been understood in 2006. Third, as with Defendants' other extrinsic evidence, this definition is inconsistent with their proposed construction. This definition recognizes that even interpreters, as defined, execute instructions as well as translating them, while the defendant's proposed construction does not include this execution function.

99. Based upon the above-described portions of the specification and prosecution history, the Court should construe "kernel" to have its plain and ordinary

meaning.

4. Single-Node Kernel

100. Claims 1, 26, 29, 35, and various dependent claims recite “single-node kernel.” I understand the parties proposed the following constructions for this claim term.

ACS Construction	NVIDIA Construction
Plain and ordinary meaning	a kernel that is designed to communicate with and run on only a single node

101. I understand Defendants’ proposed construction of “single-node kernel” to incorporate its construction of “kernel,” such that it would read:

a program code for interpreting high-level code, commands, and/or instructions supplied by a user or a script into low-level code, such as, for example, machine language or assembly language that is designed to communicate with and run on only a single node.⁶

To the extent this understanding is not consistent with the Defendants’ intention, I reserve the right to supplement this declaration.

⁶ To the extent this is not the case, a person of ordinary skill in the art would even more firmly have understood the construction of “kernel” to be its plain and ordinary meaning. If Defendants’ proposed construction of “single-node kernel” was not intended to incorporate the construction of “kernel,” then it would seemingly acknowledge that “kernel” has a plain and ordinary meaning, and that further limiting that plain and ordinary meaning in the construction of “kernel” is unnecessary as such additional limitations are provided by the remainder of the relevant claim limitations.

102. With the exception of Claim 3 (“The computer cluster of claim 1, wherein the single-node kernel comprises a Mathematica kernel.”), the claims all use “kernel” within the larger phrase, “single-node kernel.” The references to “single-node kernel” in the claims are thus discussed above. (*See above* ¶ 79.) For the same reasons as “kernel,” the numerous and varied references to a “single-node kernel” in the claims are consistent with my opinion that a person of ordinary skill in the art would have understood “single-node kernel” to have its plain and ordinary meaning.

103. A person of ordinary skill in the art would read the specification as using the term “single-node kernel” in a manner that is consistent with its plain and ordinary meaning. (*E.g.*, ’768 Patent at col. 2, ll. 58-63 (“Another embodiment provides a computer cluster that includes a plurality of nodes and a software package including a user interface and *a single-node kernel for interpreting program code instructions*. A cluster node module is configured to communicate with *the single-node kernel* and other cluster node modules.”), col. 3, ll. 14-17 (“Each node is configured to access a computer-readable medium comprising program code for a user interface and program code for *a single-node kernel module configured to interpret user instructions*.”), col. 29, l. 66-col. 30, l. 1 (“For example, *single-node kernels* can be managed using a variety of management tools and/or can be managed manually by a user, as described herein.”) (emphasis added).)

104. The specification never expressly defines “single-node kernel.” (*Id.* at col. 2, l. 58-col. 3, l. 2, col. 3, ll. 13-26, col. 29, l. 66-col. 3, l. 1.) Although some portions of the specification reference a single-node kernel (*see id.* at col. 2, l. 58-col. 3, l. 4, col. 3, ll. 13-26), a person of ordinary skill in the art would not have understood these references to a “single-node kernel” in the specification to limit the term beyond its plain and ordinary meaning.

105. The specification also identifies Mathematica as a specific single-node kernel embodiment. (*E.g., id.* at col. 3, ll. 1-2 (“In one embodiment, the single-node kernel includes a Mathematical kernel.”).) This is merely exemplary. (*See id.* at col. 4, ll. 14-18.) A person of ordinary skill in the art would not have understood this to mean that a “single-node kernel” is necessarily Mathematica, or to redefine the plain and ordinary meaning of “single-node kernel” in any manner.

106. The specification also refers to “a kernel that is designed to communicate with a single node” (*id.* at col. 1, ll. 38-43), and “a software module designed to run on a single node,” (*id.* at col. 2, ll. 18-21). (*See also id.* at col. 1, ll. 26-29; col. 4, ll. 58-59.) These sections appear to form the basis for Defendants’ proposal that single-node kernel means “a kernel that is designed to communicate with and run on only a single node.” Defendants’ proposed construction is misguided, however, because the specification never uses “designed to [communicate with / run on] a single node” in connection with the term “single-node

kernel.” A person of ordinary skill in the art would therefore not have understood this language to limit the meaning of “single-node kernel.” Moreover, these embodiments are merely exemplary embodiments.

107. Nothing in the ’768 Patent’s prosecution history suggests that “single-node kernel” should be given anything other than its plain and ordinary meaning.

108. I understand that Defendants have identified the same extrinsic evidence to support their construction of “single-node kernel” as to support their construction of “kernel.” For the reasons discussed above, a person of ordinary skill in the art would not find these reference guides, unrelated patents, and dictionaries helpful in defining “single-node kernel.” (*See above* ¶¶ 90-99.) Moreover, a person of ordinary skill in the art would have found the criticisms of Defendants’ extrinsic evidence above to apply even more strongly in this context because none of the cited portions of Defendants’ extrinsic evidence addresses the “single-node” aspect of “single-node kernel.”

109. Based upon the above-described portions of the specification and prosecution history, the Court should construe “single-node kernel” to have its plain and ordinary meaning.

5. Third Node Term

110. Claims 1, 26, and 29 recite “wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is

configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node.” I understand the parties proposed the following construction for this claim term.

ACS Construction	NVIDIA Construction
Plain and ordinary meaning	This limitation requires a precise order of operations involving three specific nodes: (1) the third node receives, from the second node, a result of a calculation that was performed by the second node in a different claim limitation; (2) the third node performs a different calculation based on the received result; and (3) the third node sends the new result to the first node.

111. In the field of cluster computing, a person of ordinary skill in the art would have understood this claim limitation to have its plain and ordinary meaning. The words of the claim limitation recite components of the third node, the evaluation of mathematical expressions by various nodes, and the movement of the results of those mathematical expression evaluations from node to node. A person of ordinary skill in the art would have understood these words to be used consistently with their plain and ordinary meanings, which would be unchanged by their use in conjunction with one another. I have reviewed the specification and prosecution history and

nothing in the intrinsic record would alter the plain and ordinary meaning of the limitation.

112. Defendants' proposal does not, in fact, construe the claim phrase, but seems rather to want to memorialize an additional comment they would like to make on the claim. Defendants' proposed construction appears to be based on a quote from a declaration I filed in support of ACS's Preliminary Patent Owner Responses in the two *inter partes* reviews involving the '768 Patent. (IPR2021-00019 Singh Declaration at 34; IPR2021-00020 Singh Declaration at 26.) In my declarations, I offered the opinion that NVIDIA Corp.'s Petitions did not show that the prior art discloses "wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node." (IPR2021-00019 Singh Declaration at 34; IPR2021-00020 Singh Declaration at 25-26.) As part of my analysis, I paraphrased the claim limitation by stating:

This limitation recites a precise order of operations involving three specific nodes: (1) the third node receives, from the second node, a result of a calculation that was performed by the second node in a different claim limitation; (2) the third node performs a different calculation based on the received result; and (3) the third node sends the new result to the first node.

(IPR2021-00019 Singh Declaration at 34; IPR2021-00020 Singh Declaration at 26.)

113. A person of ordinary skill in the art would not have understood my statement to redefine the claim limitation or otherwise alter the plain and ordinary meaning of the phrase. Although there were several terms for which I offered an opinion as to claim construction, this was not one of them. When offering a construction, I was clear. (*See, e.g.*, IPR2021-00019 Singh Declaration at 21 (“The Board should construe ‘peer-to-peer architecture’”), 23 (“the Board should construe ‘cluster node module’”).) In contrast, my statement above paraphrased certain portions of the third node claim limitation to provide context for my validity analysis, nothing more.

114. Placing Defendants’ proposed construction into the claims illustrates that that it is mere commentary, and not suitable as a construction. For example, Claim 1 in the relevant part read would read:

1. A computer cluster comprising:
 - a plurality of nodes . . .
 - wherein the plurality of nodes comprises:
 - a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and
 - a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of the first

mathematical expression evaluation to a third node;

[t]his limitation requires a precise order of operations involving three specific nodes: (1) the third node receives, from the second node, a result of a calculation that was performed by the second node in a different claim limitation; (2) the third node performs a different calculation based on the received result; and (3) the third node sends the new result to the first node;

(text from Defendants’ proposed construction underlined and in red.) Defendants’ proposed construction is impossible to reconcile with the language of the claim. Plainly, a limitation cannot self-referentially read “[t]his limitation requires a precise order of operations involving three specific nodes,” and expressly refer to other limitations by stating “a result of a calculation that was performed by the second node in a different claim limitation.”

115. Based upon the above-described portions of the specification and prosecution history, the Court should construe “wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node” to have its plain and ordinary meaning.

6. A mechanism for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture

a. The limitation recites sufficiently definite structure to perform the claimed function

116. Claim 1 recites “a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture.” Claim 35 recites “a mechanism to communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture.” The claim language used in Claim 35 is similar to the language used in Claim 1. A comparison of the claim language is below:

Claim 1	Claim 35
“a mechanism <u>for the nodes</u> to communicate results of <u>mathematical expression</u> evaluation with <u>each other</u> using a peer-to-peer architecture.”	“a mechanism to communicate results of evaluation with <u>other computer cluster nodes</u> using a peer-to-peer architecture.”

(emphasis added). I address Claims 1 and 35 together due to the similarity in the claim language.

117. I understand that the parties dispute whether the claim term is subject to 35 U.S.C. § 112, ¶ 6, *i.e.*, whether it is a means-plus-function limitation.

ACS Construction	NVIDIA Construction
Not subject to 35 U.S.C. § 112, ¶ 6; no construction necessary.	Subject to § 112, ¶ 6

<p>Should the Court determine that § 112, ¶ 6 applies:</p> <p><u>Function (Claim 1):</u> communicate results of mathematical expression evaluation and user instructions between nodes</p> <p><u>Function (Claim 35):</u> communicate results of evaluation and user instructions between nodes</p> <p><u>Structure:</u> messaging modules that support communication using a peer-to-peer architecture</p>	<p><u>Function:</u> communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture</p> <p><u>Structure:</u> Message-Passing Interface (“MPI”) module 302 RMQ received message queue 306 and MRQ message receiving queue 308 configured to execute the process described at 25:29-43.</p>
--	---

118. A person of ordinary skill in the art would have understood the “mechanism” limitation to recite sufficient structure when considered in the context of the claims as a whole. The word mechanism is used several times in Claims 1 and 35, and in dependent claims:

Claim	Language ⁷
1	<p>“<u>a mechanism</u> for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture”</p> <p>“after accepting user instructions, communicate at least some of the user instructions using <u>the mechanism</u> for the nodes to communicate with each other”</p> <p>“after communicating at least some of the user instructions using <u>the</u></p>

⁷ Emphasis added to highlight the use of the word “mechanism” in the claims.

	<u>mechanism</u> , communicate at least some of the user instructions to one or more single-node kernels”
35	<p>“<u>a mechanism</u> to communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture”</p> <p>“after accepting user instructions, communicate at least some of the user instructions using <u>the mechanism</u> for the nodes to communicate with each other”</p> <p>“after communicating at least some of the user instructions using <u>the mechanism</u>, communicate at least some of the user instructions to the single-node kernel”</p>
3	“[t]he computer cluster of claim 1, wherein <u>the mechanism</u> comprises a message passing interface.”
21	“The computer cluster of claim 20, wherein one or more of the nodes are configured to communicate at least some of the user instructions using <u>the mechanism</u> for the nodes to communicate with each other.”

119. For each of Claims 1 and 35, the first limitation containing “a mechanism” provides the antecedent basis for the subsequent limitations containing “the mechanism.” Claims 1 and 35 therefore recite that the mechanism is for the nodes to “communicate results of mathematical expression evaluation” and to “communicate at least some of the user instructions” between nodes. The claims also recite “peer-to-peer architecture,” which provides additional explicit structure by identifying the relative peer-to-peer relationship of the nodes communicating via the mechanism.⁸ This claim language connotes structure to a person of ordinary skill

⁸ I discussed the structure of a peer-to-peer architecture in more detail above. (See above ¶¶ 47-63.)

in the art. A peer-to-peer architecture is a structure, and a mechanism to communicate over a peer-to-peer architecture would be understood to invoke hardware and/or software modules that enable communication via messaging.

120. Dependent Claim 3 adds additional structure by reciting that “the mechanism comprises a message passing interface.” As a dependent claim, the structure identified in Claim 3 does not limit the structure in Claim 1, but it supports my opinion that the limitation in Claim 1 provides sufficiently definite structure. The mechanism introduced in Claim 1 is necessarily broader than that in Claim 3, which further comprises a message passing interface. A message passing interface is consistent with my understanding that Claim 1 would be understood to invoke hardware and/or software modules that enable communication via messaging.

121. The specification describes examples of the mechanism for the nodes to communicate with each other using a peer-to-peer architecture. The specification depicts one embodiment of a peer-to-peer architecture in FIG. 2, below:

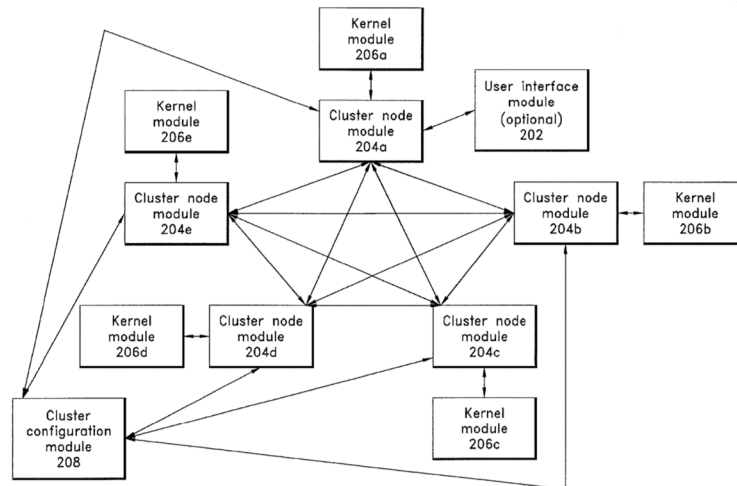


FIG. 2

122. The specification discloses a messaging module that supports communication using a peer-to-peer architecture. For example:

As shown in FIG. 2, each of the cluster node modules 204a-e is in communication with each of the other cluster node modules 204a-e. For example, one cluster node module 204a is in communication with all of the other 45 cluster node modules 204b-e.

....

The first cluster node module 204a is in communication with one or more other cluster node modules 204b, 204c, and so forth, each of which provides a set of MPI calls and/or advanced cluster commands. In one embodiment, MPI may be used to send messages between nodes in a computer cluster. Communications can occur between any two or more cluster node modules (for example, between a cluster node module 204a and another cluster node module 204c) and not just between “adjacent” kernels.

(’768 Patent at col. 5, l. 56-col. 6, l. 13.)

123. The specification discloses an “MPI module” as one optional component of the cluster node module. For example:

FIG. 3 shows one embodiment of a cluster node module 204 implementing MPI calls and advanced MPI functions. In the

embodiment shown in FIG. 3, cluster node module 204 includes MPI module 302, advanced functions module 304, received message queue 306, and message receiving queue 308.

(*Id.* at 12:41-46 (emphasis added); *see generally id.* at col. 12, l. 47-col. 16, l. 39; *see, e.g., id.* at col. 13, ll. 15-18 (“[B]asic MPI calls include calls that send data, equations, formulas, and/or other expressions. Simply sending expressions from one node to another is possible with these most basic MPI calls.”), col. 14, ll. 37-41 (“[C]ollective MPI calls . . . provide basic multi-node data movement across nodes[. Collective MPI calls can include broadcasts, gathers, transpose, and other vector and matrix operations, for example.”).)

124. The specification explicitly identifies the MPI module implementation of collective calls as one example of the claimed mechanism. (*See id.* at col. 14, ll. 35-43 (“Collective calls can also provide commonly used ***mechanisms*** to send expressions between groups of nodes.”) (emphasis added).) By associating the MPI module implementation of collective calls with the mechanism, the specification confirms that the claim limitation “a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture” provides a sufficiently definite meaning as the name for the structure that performs the function. The MPI module implementation of collective calls is one example discussed in the specification, but not the only structure a person of ordinary skill in the art would have understood the claim to invoke.

125. The specification also discloses an exemplary process that uses the mechanism for the nodes to communicate with each other using a peer-to-peer architecture:

B. *Cluster Node Module Operation*

In one embodiment, a cluster node module 204 implements cluster computing support for a kernel module 206 during a main loop, as shown in FIG. 5.

. . . .

In one embodiment, when a user enters a command in the user interface module 202, the cluster node module 204a connected to the user interface module 202 submits the user command to all other cluster node modules 204b-e in the computer cluster 100. The user commands can be simple (for example, “1+1”), but can also be entire subroutines and sequences of code (such as, for example, Mathematica code), including calls to MPI from within the user interface module 202 (for example, the Mathematica Front End) to perform message passing between kernel modules 206a-e (for example, Mathematica kernels). These include the fundamental MPI calls, which are implemented using specially identified messages between a cluster node module 204 and its local kernel module 206.

. . . .

In one embodiment, the cluster node module 204a connected to the user interface module 202 submits the user command to the kernel module 206a to which it is connected. Each of the other cluster node modules 204b-e, after receiving the message, submits the command to the respective kernel module 206b-e to which it is connected.

. . . .

Depending on the nature of the result from the kernel module, the cluster node module 204 can report the result to a local computer system or pass the result as a message to another cluster node module 204.

. . . .

Messages from other cluster node modules 204 are responded to at 512. In one embodiment, each cluster node module (for example, the cluster node module 204a) checks for and responds to messages from other cluster node modules 204b-e This process provides the peer-to-peer behavior of the cluster node modules 204a-e. *Via this*

mechanism, code running within multiple, simultaneously running kernel modules (for example, Mathematica kernels) ***can interact on a pair-wise or collective basis***, performing calculations, processing, or other work on a scale larger and/or faster than one kernel could have done alone. In this manner, ***user-entered instructions and data specifying what work will be done via user commands can be executed more quickly and/or reliably***.

(*Id.* at col. 24, l. 35-col. 25, l. 47 (emphasis added).) A person of ordinary skill in the art would have understood this disclosure to describe an exemplary messaging operation of the cluster node. This disclosure again equates the mechanism with the use of software or hardware used to implement messaging for communication between nodes using a peer-to-peer architecture.

126. The prosecution history supports my understanding that a person of ordinary skill in the art would have understood the claim term to connote sufficiently definite structure to perform the recited function. Claim 1 as originally filed recited:

2. (New) A computer cluster comprising:
 - a plurality of nodes, wherein one or more of the nodes receives a command to start a cluster initialization process for the computer cluster, and wherein each of the nodes is configured to access a computer-readable medium comprising program code for a single-node kernel module configured to interpret user instructions; and
 - a mechanism for the nodes to communicate with each other.

(September 8, 2014 Preliminary Amendment at 2.)⁹

⁹ References to the prosecution of Claim 1 should be understood to refer to the prosecution of draft Claim 2, which was renumbered as Claim 1 prior to issuance. (November 29, 2018 Issue Classification at 3.)

127. During prosecution the Examiner concluded that the limitation containing “module configured to” invoked 35 U.S.C. § 112, ¶ 6. (July 20, 2016 Non-Final Rejection at 4-7.) The Examiner *did not* similarly identify the limitation “a mechanism for the nodes to communicate with each other . . .” as invoking 35 U.S.C. § 112, ¶ 6. (*Id.*) The Examiner’s action of invoking 35 U.S.C. § 112, ¶ 6 to construe a “module configured to,” while choosing not to invoke 35 U.S.C. § 112, ¶ 6 to construe the limitation “a mechanism for” is consistent with my opinion that a person of ordinary skill in the art would have understood the claim term to connote sufficiently definite structure to perform the recited function.

128. The Examiner also rejected Claim 1 under 35 U.S.C. § 101 as being directed to non-statutory subject matter, *i.e.*, the Examiner thought the claim was directed to an abstract idea. (July 20, 2016 Non-Final Rejection at 7-13.) In response, ACS amended the “mechanism for the nodes to communicate with each other” limitation to add “using a peer-to-peer architecture” as follows:

2. (Currently amended) A computer cluster comprising:
 - a plurality of nodes comprising a hardware processor, wherein one or more of the nodes receives a command to start a cluster initialization process for the computer cluster, and wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, causes the hardware processor ~~module configured~~ to interpret user instructions; and
 - a mechanism for the nodes to communicate with each other using a peer-to-peer architecture;
 - wherein at least one of the nodes returns a result to a user interface or a script.

(November 21, 2016 Response to Office Action Dated July 20, 2016 at 2 (highlighting added); *see also* November 11, 2016 Applicant-Initiated Interview Summary.)

129. ACS argued that the amendment overcame the non-statutory subject matter rejection because “using a peer-to-peer architecture” claimed improvements to the functioning of the computer itself. The claim was thus directed at more than an abstract idea. (November 21, 2016 Response to Office Action Dated July 20, 2016 at 8-9.)

130. The Examiner withdrew the subject matter eligibility rejection based on the amendment. (March 9, 2017 Final Rejection at 3). The Examiner’s recognition of “a mechanism for the nodes to communicate with each other using a peer-to-peer architecture” as a tangible improvement to the operation of the computer that is not directed to an abstract idea is consistent with my opinion that the claimed “peer-to-peer architecture” provides additional structure to the claim.

131. NVIDIA Corp. also filed two petitions seeking *inter partes* review of Claims 1 and 35 of the ’768 Patent. NVIDIA Corp. submitted two 78-page petitions and a 10-page reply, but it did not identify any of the claim limitations for construction under 35 U.S.C. § 112, ¶ 6. (IPR2021-00019 Petition for *Inter Partes* Review at 14; IPR2021-00020 Petition for *Inter Partes* Review at 16-17; *see* IPR2021-00019 and IPR2021-00020 Petitioner’s Reply.) NVIDIA Corp. also

submitted two declarations from its expert, Dr. Henry Tufo. Dr. Tufo testified that his background and expertise qualified him as an expert in the technical issues of the '768 Patent. (IPR2021-00019 Tufo Declaration at 2-5; IPR2021-00020 Tufo Declaration at 1-5.) He also testified that he analyzed the challenged claims “by applying their plain and ordinary meaning as they would have been understood by POSITA at the time of the invention.” (IPR2021-00019 Tufo Declaration at 24; IPR2021-00020 Tufo Declaration at 25.) Dr. Tufo went on to opine on the limitation “a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture,” stating that:

The claimed mechanism is disclosed in the Distributed Maple Publications as the combination of two software components: dist.Scheduler, which “coordinates node interaction,” and dist.maple, which “implements the interface between kernel and scheduler.” This scheduler-based message passing mechanism is installed on each node and is used to communicate results of mathematical expression evaluation between nodes using a peer-to-peer architecture: “A session comprises of a set of nodes each of which holds a pair of processes: a kernel and a scheduler.”

(IPR2021-00019 Tufo Declaration at 40 (internal citations omitted); *see also* IPR2021-00020 Tufo Declaration at 77 (similar analysis of Claim 35 language).) Dr. Tufo’s ability to understand the claimed structure, and to analyze whether the prior art disclosed the claimed structure, is consistent with my opinion that a person of ordinary skill in the art would read Claims 1 and 35 to recite sufficiently definite structure to perform the recited function.

132. Accordingly, the intrinsic evidence supports my opinion that a person of ordinary skill in the art reading the specification would have understood the claim terms to have a sufficiently definite meaning as the name for the structure that performs the function.

b. Alternative function and corresponding structure

133. I have also been instructed to provide my opinion on the appropriate function and structure if the Court determines that § 112, ¶ 6 applies to this limitation.

134. It is my opinion that a person of ordinary skill in the art would have understood Claim 1 to recite the function: “communicate results of mathematical expression evaluation and user instructions between nodes” and Claim 35 to recite the similar function: “communicate results of evaluation and user instructions between nodes.”

135. “[C]ommunicate results of [mathematical expression] evaluation” is plainly stated in the limitations at issue. Because the first use of “a mechanism” provides the antecedent basis for the claim’s subsequent uses of “the mechanism,” any function associated with later recitations of “the mechanism” must also be part of the function. “[C]ommunicate . . . user instructions” is plainly stated in subsequent limitations in association with “the mechanism.”

136. As discussed above a person of ordinary skill in the art would have

understood “using a peer-to-peer architecture” to describe a structure for communicating between the nodes. (*See above* ¶¶ 47-63.) Therefore, an architecture, such as a peer-to-peer architecture, is structure, not a function.

137. Defendants’ proposed function is incorrect. First, communicating user instructions between nodes is absent from Defendants’ proposed construction. Second, Defendants’ proposed construction improperly includes “using a peer-to-peer architecture” as part of the function.

138. It is my opinion that a person of ordinary skill in the art would have understood the corresponding structure for both Claims 1 and 35 to be: “messaging modules that support communication using a peer-to-peer architecture.”

139. The specification clearly links messaging modules that support communication using a peer-to-peer architecture with the function. (’768 Patent at col. 3, ll. 46-49 (“The first node module, the second node module, and the third node module are configured to communicate with one another using a peer-to-peer architecture.”).) The specification distinguishes this from a master slave architecture. (*Id.* at col. 12, ll. 33-37 (“computer cluster having peer-to-peer node architecture performs computations that can be more efficient, easier to design, and/or more reliable than similar computations performed on grid computers having master-slave node architecture.”).)

140. The specification discloses using the messaging modules that support

communication using a peer-to-peer architecture to communicate results of mathematical expression evaluations. For example, the specification discloses “[r]esults of evaluations performed by kernel modules 206a-e are communicated back to the first cluster node module 204a via the cluster node modules 204a-e, which communicates them to the user interface module 208. (*Id.* at col. 6, ll. 22-25.) The specification also discloses

At 510, a cluster node module 204 receives a result from a kernel module 206. In one embodiment, once the kernel module 206 completes its evaluation, it returns the kernel module's output to the cluster node module 204 to which it is connected. Depending on the nature of the result from the kernel module, the cluster node module 204 can report the result to a local computer system or pass the result as a message to another cluster node module 204.

...

Messages from other cluster node modules 204 are responded to at 512. In one embodiment, each cluster node module (for example, the cluster node module 204a) checks for and responds to messages from other cluster node modules 204b-e and from the kernel module 206a repeatedly until those are exhausted.

(*Id.* at col. 26, ll. 9-27.) In these sections, results of mathematical expression evaluations are communicated between nodes using messaging between the cluster node modules, which allow exchanging messages among nodes using a peer-to-peer architecture as discussed above. (*Id.* at col. 12, ll. 48-51, col. 14, ll. 26-39, col. 15, ll. 52-55, col. 16, ll. 41-42.)

141. The specification identifies exemplary preferred embodiments of messaging modules that supports communication using a peer-to-peer architecture.

For example, the specification discloses a cluster node module that supports communication using a peer-to-peer architecture. (*Id.* at col. 25, ll. 22-38; *see also id.* at col. 2, ll. 30-56 (“The second cluster node module . . . communicates with the first cluster node module. . . . The third cluster node module is configured to communicate with the first cluster node module and the second cluster node module”), col. 3, ll. 17-24 (“The cluster includes a plurality of cluster node modules. Each cluster node module is configured to communicate with . . . one or more other cluster node modules, . . . such that the plurality of cluster node modules communicate with one another in order to act as a cluster.”), col. 4, ll. 61-62 (“In one embodiment, cluster node modules communicate with the kernels and with each other in order to implement cluster computing functionality.”), col. 6, ll. 10-13 (“Communications can occur between any two or more cluster node modules (for example, between a cluster node module 204 a and another cluster node module 204 c) and not just between ‘adjacent’ kernels.”).)

142. In one embodiment the cluster node module includes an optional MPI module within the cluster node module that supports communication using a peer-to-peer architecture. (*See id.* at col. 12, l. 41-col. 16, l. 39, col. 11, ll. 42-45.)

143. The specification also identifies particular types of calls (*e.g.*, basic or collective calls) that support communication using a peer-to-peer architecture. For example, the specification states

In one embodiment, the MPI module 302 can include basic MPI calls such as, for example, relatively low-level routines that map MPI calls that are commonly used in other languages (such as C and Fortran), so that such calls can be available directly from the Mathematica user interface 204. In some embodiments, basic MPI calls include calls that send data, equations, formulas, and/or other expressions.).

(*Id.* at col. 13, ll. 10-16.). The specification also states “[i]n one embodiment, the MPI module 302 can include program code for implementing collective MPI calls (for example, calls that provide basic multi-node data movement across nodes).” (*Id.* at col. 14, ll. 36-39.)

144. The specification also identifies an optional advanced functions module within the cluster node module that supports communication using a peer-to-peer architecture in the context of implementing more complex communication functionality. (*See id.* at col. 16, l. 40-col. 21, l. 10.)

145. A person of ordinary skill in the art would not have understood the structure to be limited to any of the exemplary preferred embodiments because the specification discusses the messaging between cluster node modules both generally and in association with these exemplary preferred embodiments. A person of ordinary skill in the art would have understood the general references to refer to the broader class of messaging modules that support communication using a peer-to-peer architecture. Additionally, I am informed that when construing a means-plus-function claim, the structure should embody the equivalents to the structure disclosed in the specification as performing the function. “Messaging modules that

support communication using a peer-to-peer architecture” is also equivalent to the exemplary embodiments disclosed in the specification.

146. Defendants’ corresponding structure for this limitation is incorrect. First, their structure identifies portions of a preferred embodiment of the cluster node module as the structure, including MPI module 302, RMQ received message queue 306 and MRQ message receiving queue 308. This is a very specific identification of structure that ignores other structure identified in the specification that is linked with performing the claimed function, and does not provide for equivalents.

147. Moreover, portions of Defendants’ proposed structure is not required for performing the recited function. Defendants’ proposed structure includes the message receiving queue (“MRQ”) and received message queue (“RMQ”). These queues are described in connection with asynchronous calls, but not with peer-to-peer communications between the nodes generally. (*Compare id.* at col. 13, l. 41-col. 14, ll. 28 (asynchronous calls) *with id.* at col. 13, ll. 9-40 (basic MPI calls), col. 14, l. 35-col. 15, l. 50 (collective MPI calls), col. 15, l. 51-col. 16, l. 20 (MPI communicator calls), col. 16, ll. 21-38 (other MPI support calls); *see also id.* at col. 25, ll. 31-37.)

148. Accordingly, it is my opinion that a person of ordinary skill in the art would have understood the claimed function as “communicate results of mathematical expression evaluation and user instructions between nodes” and the

corresponding structure as “messaging modules that support communication using a peer-to-peer architecture.”

7. A mechanism for the nodes to communicate results of mathematical expression evaluation with each other

a. The limitation recites sufficiently definite structure to perform the claimed function

149. Claim 29 recites “a mechanism for the nodes to communicate results of mathematical expression evaluation with each.” The claim language used in Claim 29 is similar to the language used in Claims 1, discussed above.

150. I understand that the parties dispute whether the claim term is subject to 35 U.S.C. § 112, ¶ 6, *i.e.*, whether it is a means-plus-function limitation.

ACS Construction	NVIDIA Construction
<p>Not subject to 35 U.S.C. § 112, ¶ 6; no construction necessary.</p> <p>Should the Court determine that § 112, ¶ 6 applies:</p> <p><u>Function:</u> communicate results of mathematical expression evaluation and user instructions between nodes</p> <p><u>Structure:</u> messaging modules that support communication using a peer-to-peer architecture</p>	<p>Subject to § 112, ¶ 6</p> <p><u>Function:</u> communicate results of mathematical expression evaluation with each other”</p> <p><u>Structure:</u> Message-Passing Interface (“MPI”) module 302 RMQ received message queue 306 and MRQ message receiving queue 308 configured to execute the process described at 25:29-43.</p>

151. For the same reasons I discussed above with respect to the similarly

worded limitation of Claim 1, the language of Claim 29, the specification, and the prosecution history support my understanding that a person of ordinary skill in the art would have understood Claim 29 to connote sufficiently definite structure to perform the recited function. (*See above* ¶¶ 116-132.) Claim 29 includes similar claim language to Claims 1 and 35. Although Claim 29 does not recite a peer-to-peer architecture in the limitation, as discussed above the peer-to-peer architecture is a feature of the communication described in the specification (*see above* ¶¶ 51-55, 116-132). Thus, Claim 29 connotes the same structure as in Claim 1.¹⁰ The same sections of the specification that disclose and elaborate upon the structure provided in Claim 1 also disclose and elaborate upon the structure provided in Claim 29. The Examiner also never identified “a mechanism” in Claim 29 as subject to 35 U.S.C. § 112, ¶ 6. The Examiner did not reject Claim 29 under 35 U.S.C. § 101 as being directed to non-statutory subject matter, *i.e.*, for claiming an abstract idea. NVIDIA Corp.’s *inter partes* review expert was again able to understand the claimed structure, and to analyze whether the prior art disclosed the claimed structure

152. NVIDIA includes the same structure in its proposed construction for Claim 29 that it included in its proposed constructions for both Claim 1 and 35.

¹⁰ Defendants appears to agree on this point because Defendants’ proposed construction for Claims 1, 29 and 35 includes the same structure for all three, despite some difference in the claim language.

Although I disagree with NVIDIA’s proposed construction, I do agree that a person of ordinary skill in the art would have understood Claim 29 to connote the same structure as Claim 1. Claim 29 would be understood to invoke hardware and/or software modules that enable communication via messaging.

153. The word mechanism is used several times in Claim 29:

Claim	Language ¹¹
29	<p>“<u>a mechanism</u> for the nodes to communicate results of mathematical expression evaluation with each other”</p> <p>“after accepting user instructions, communicate at least some of the user instructions using <u>the mechanism</u> for the nodes to communicate with each other”</p> <p>“after communicating at least some of the user instructions using <u>the mechanism</u>, communicate at least some of the user instructions to one or more single-node kernels”</p>

154. For the same reasons I discussed above with respect to the similarly worded limitation of Claim 1, the prosecution history supports my understanding that a person of ordinary skill in the art would have understood the claim term to connote sufficiently definite structure to perform the recited function. (*See above* ¶¶ 116-132.) The Examiner never identified “a mechanism for the nodes to communicate results of mathematical expression evaluation with each other” in Claim 29 as subject to 35 U.S.C. § 112, ¶ 6. The Examiner did not reject Claim 29

¹¹ Emphasis added to highlight the use of the word “mechanism” in the claims.

under 35 U.S.C. § 101 as being directed to non-statutory subject matter, *i.e.*, for claiming an abstract idea. NVIDIA Corp. filed a petition seeking *inter partes* review of Claim 29 and did not identify any of the limitations of Claim 29 for construction under 35 U.S.C. § 112, ¶ 6. NVIDIA Corp.’s *inter partes* review expert Dr. Tufo was again able to understand the claimed structure, and to analyze whether the prior art disclosed the claimed structure.

b. Alternative function and corresponding structure

155. It is my opinion that a person of ordinary skill in the art would have understood this limitation to recite the function: “communicate results of mathematical expression evaluation and user instructions between nodes.” The claim language used in Claim 29 is similar to the language used in Claims 1 and 35. For the same reasons I discussed with respect to Claims 1 and 35, a person of ordinary skill in the art would interpret the function to include communicating results of mathematical expression evaluation and user instructions. (*See above* ¶¶ 133-136.) Defendants’ proposed construction again omits the user instructions from the function.

156. If the Court determines that this limitation is a means-plus-function limitation, it is my opinion that a person of ordinary skill in the art would have understood the corresponding structure to be: “messaging modules that support communication using a peer-to-peer architecture.”

157. For the same reasons I discussed above with respect to the similarly worded limitation of Claims 1 and 35, a person of ordinary skill in the art would have understood this the corresponding structure recited by the specification to perform the claimed function to be the structure identified by ACS and not the one identified by Defendants. (*See above* ¶¶ 138-148.) Although Claim 29 does not recite a peer-to-peer architecture in the limitation, as discussed above the peer-to-peer architecture is a feature of the communication described in the specification (*see above* ¶¶ 51-55, 116-132). This identification of the same corresponding structure for these limitations in Claims 1 and 29 is supported by Defendants' identification of the identical structure for the two limitations, albeit the incorrect structure.

8. A mechanism for the nodes to communicate results of mathematical expression evaluation with each other using asynchronous calls

a. The limitation recites sufficiently definite structure to perform the claimed function

158. Claim 26 recites “a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using asynchronous calls.” The claim language used in Claim 26 is similar to the language used in Claim 1, discussed above. A comparison of the claim language is below

Claim 1	Claim 26
“a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using <u>a peer-to-peer architecture</u> .”	“a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using <u>asynchronous calls</u> .”

(emphasis added).

159. I understand that the parties dispute whether the claim term is subject to 35 U.S.C. § 112, ¶ 6, *i.e.*, whether it is a means-plus-function limitation.

ACS Construction	NVIDIA Construction
<p>Not subject to 35 U.S.C. § 112, ¶ 6; no construction necessary.</p> <p>Should the Court determine that § 112, ¶ 6 applies:</p> <p><u>Function</u>: communicate results of mathematical expression evaluation and user instructions between nodes</p> <p><u>Structure</u>: messaging modules that support asynchronous communication using a peer-to-peer architecture</p>	<p>Subject to § 112, ¶ 6</p> <p><u>Function</u>: communicate results of mathematical expression evaluation with each other using asynchronous calls</p> <p><u>Structure</u>: Message-Passing Interface (“MPI”) module 302, including mpiSend, mpiRecv, and mpiTest commands as described at 14:1-28, RMQ received message queue 306 and MRQ message receiving queue 308 configured to execute the process described at 25:29-43.</p>

160. A person of ordinary skill in the art would have understood the “mechanism” limitation to recite sufficient structure when considered in the context of the claims as a whole. The word mechanism is used several times in claim 26.

Claim	Language ¹²
26	<p>“<u>a mechanism</u> for the nodes to communicate results of mathematical expression evaluation with each other using asynchronous calls”</p> <p>“after accepting user instructions, communicate at least some of the user instructions using <u>the mechanism</u> for the nodes to communicate with each other”</p> <p>“after communicating at least some of the user instructions using <u>the mechanism</u>, communicate at least some of the user instructions to one or more single-node kernels”</p>

161. Claim 26’s first limitation containing “a mechanism” provides the antecedent basis for the subsequent limitations in Claim 26 containing “the mechanism.” Claim 26 therefore recites that the mechanism is for the nodes to “communicate results of mathematical expression evaluation with each other” and to communicate “at least some of the user instructions” between nodes. Claim 26 further recites that the mechanism uses “asynchronous calls,” which provides additional explicit structure by identifying the asynchronous nature of the communication between nodes via the mechanism. Although Claim 26 does not also recite a peer-to-peer architecture in the limitation, as discussed above the peer-to-peer architecture is a feature of the communication described in the specification (*see above* ¶¶ 51-55, 116-132). A peer-to-peer architecture is a structure as are asynchronous calls, and a mechanism to communicate using asynchronous calls

¹² Emphasis added to highlight the use of the word “mechanism” in the claims.

would be understood to invoke hardware and/or software modules that enable asynchronous communication.

162. Dependent Claims 27, 28, and 30 add additional structure. Claim 27 recites

27. The computer cluster of claim 26, wherein the asynchronous calls comprise a first command to create a first packet containing:
an expression to be sent as payload; and
a target node where the expression should be sent;
wherein the first command is configured to be called from within a single-node kernel;
wherein the single-node kernel is configured to send the first packet to a local cluster node module; and
wherein the local cluster node module is configured to forward the expression to the target node.

(’768 Patent at cl. 27.) Claim 28 recites

28. The computer cluster of claim 27, wherein the asynchronous calls comprise a second command to create a second packet containing:
a location where the expression is expected to be received; and
a sending node from which the expression is expected to be received;
wherein the second command is configured to be called from within a single-node kernel;
wherein the single-node kernel is configured to send the second packet to a local cluster node module; and
wherein the local cluster node module is configured to store the second packet contents in a message receiving queue.

(*Id.* at cl. 28.) And Claim 30 recites

30. The computer cluster of claim 4, wherein each of the plurality of nodes implements asynchronous calls that enable the single-node kernel to perform computation tasks while the cluster node modules are simultaneously communicating with one another.

(*Id.* at cl. 30.)

163. As dependent claims, the structure identified in these dependent claims does not limit the structure recited in Claim 26, but it supports my opinion that the limitation in Claim 26 provides a sufficiently definite structure. The mechanism introduced in Claim 26 is necessarily broader than that in the dependent claims, which provide further details about the content and handling of the asynchronous calls. These details are consistent with the understanding that Claim 26 would be understood to invoke hardware and/or software modules that enable asynchronous communication.

164. In addition to the portions of the specification that I addressed with respect to Claim 1 (*see above* ¶¶ 121-125), the specification discusses asynchronous calls.

165. A person of ordinary skill in the art would find additional support for the structure provided by the reference to asynchronous calls in the specification. The specification discloses an exemplary embodiment of asynchronous calls:

Asynchronous MPI Calls

Asynchronous calls make it possible for the kernel to do work while communications are proceeding simultaneously. It is also possible that another node may not be able to send or receive data yet, allowing one kernel to continue working while waiting.

TABLE C

Call	Description
<code>mpiISend[expr, target, comm, tag, req]</code>	Sends an expression <code>expr</code> to a processor with the ID <code>target</code> in the communicator <code>world comm</code> , returning immediately. It can be balanced with calls to <code>mpiTest[req]</code> until <code>mpiTest[req]</code> returns True.
<code>mpiIRecv[expr, target, comm, tag, req]</code>	Receives an expression <code>expr</code> from a processor with the ID <code>target</code> in the communicator <code>world comm</code> , returning immediately. It can be balanced with calls to <code>mpiTest[req]</code> until <code>mpiTest[req]</code> returns True. The <code>expr</code> is not safe to access until <code>mpiTest[req]</code> returns True.
<code>mpiTest[req]</code>	Completes asynchronous behavior of <code>mpiISend</code> and <code>mpiIRecv</code>
<code>mpWait[req]</code>	Calls <code>mpiTest</code> until it returns True.
<code>mpiWaitall[reqlist]</code>	Calls <code>mpiWait</code> all on every element of <code>reqlist</code>
<code>mpiWaitany[reqlist]</code>	Calls <code>mpiTest</code> on each element of <code>reqlist</code> until one of them returns True

(’768 Patent at col. 13, ll. 41-66.)

166. The specification also provides the following example and description of the cluster node module’s asynchronous MPI calls:

The `mpiISend` and `mpiIRecv` commands have a letter “I” to indicate asynchronous behavior, making it possible to do other work while messages are being sent and received, or if the other processor is busy. So, the above example could be done asynchronously:

```
In[7]:= If[1==Mod[$IdProc, 2],mpiISend[N[Pi,22],targetProc,
      mpiCommWorld,d,e],
      mpiIRecv[a,targetProc,mpiCommWorld,d,e]]
```

(*Id.* at col. 27, ll. 38-48.)

167. A person of ordinary skill in the art would have understood this disclosure to describe one embodiment of the types of asynchronous messaging that would be part of the claimed mechanism. The asynchronous calls disclosed in the

specification illustrate one example of the messaging implemented by the mechanism, but not the only structure a person of ordinary skill in the art would have understood the claim to invoke.

168. For the same reasons I discussed above with respect to the similarly worded limitation of Claim 1, the prosecution history supports my understanding that a person of ordinary skill in the art would have understood Claim 26 to connote sufficiently definite structure to perform the recited function. (*See above* ¶¶ 116-132.) The Examiner never identified “a mechanism” in Claim 26 as subject to 35 U.S.C. § 112, ¶ 6. The Examiner did not reject Claim 26 under 35 U.S.C. § 101 as being directed to non-statutory subject matter, *i.e.*, for claiming an abstract idea. NVIDIA Corp.’s *inter partes* review expert was again able to understand the claimed structure, and to analyze whether the prior art disclosed the claimed structure

169. Accordingly, the intrinsic evidence supports my opinion that a person of ordinary skill in the art reading the specification would have understood the claim term to have a sufficiently definite meaning as the name for the structure that performs the function.

b. Alternative function and corresponding structure

170. It is my opinion that a person of ordinary skill in the art would have understood this limitation to recite the function: “communicate results of mathematical expression evaluation and user instructions between nodes.” The

claim language used in Claim 26 is similar to the language used in Claim 1. For the same reasons I discussed with respect to Claim 1, a person of ordinary skill in the art would interpret the function to include communicating results of mathematical expression evaluation and user instructions. (*See above* ¶¶ 133-136.) Like with peer-to-peer architecture, a person of ordinary skill in the art would have understood “asynchronous calls” to describe a structure for communicating between the nodes. (*See above* ¶¶ 158-169.) Therefore, “the use of asynchronous calls” is not part of the function. Defendants’ proposed construction again omits the user instructions from the function and erroneously includes asynchronous calls.

171. If the Court determines that this limitation is a means-plus-function limitation, it is my opinion that a person of ordinary skill in the art would have understood the corresponding structure to be: “messaging modules that support asynchronous communication using a peer-to-peer architecture.”

172. This limitation differs from the corresponding limitation in Claim 1 by reciting a different structure to carry out the claimed function: asynchronous calls. Although Claim 26 does not recite a peer-to-peer architecture in the limitation, as discussed above the peer-to-peer architecture is a feature of the communication described in the specification (*see above* ¶¶ 51-55, 116-132).

173. The specification identifies exemplary preferred embodiments of messaging modules that supports asynchronous communication using a peer-to-peer

architecture. For example, the specification discloses a cluster node module that supports asynchronous communication. ('768 Patent at col. 6, ll. 19-21 (“The cluster node modules 204a-e provide communications among kernel modules 206a-e while the tasks are executing.”).) The specification identifies types of asynchronous calls using a peer-to-peer architecture. (*Id.* at col. 13, l. 41-col. 14, l. 34.) The specification also identifies an advanced functions module within the cluster node module that supports asynchronous communication using a peer-to-peer architecture in the context of implementing more complex communication functionality. (*Id.* at col. 16, l. 39-col. 20, l. 25.)

174. For some of the reasons I discussed above with respect to Claim 1, a person of ordinary skill in the art would have understood Defendants’ identified structure to be overly narrow in light of the specification’s disclosure of exemplary embodiments and to exclude their equivalents. (*See above* ¶¶ 138-148.)

175. Accordingly, it is my opinion that a person of ordinary skill in the art would have understood the claimed function as “communicate results of mathematical expression evaluation and user instructions between nodes” and the associated structure as “messaging modules that support asynchronous communication using a peer-to-peer architecture.”

9. First Node Terms

176. Claim 35 recites “wherein the third node is configured to receive the

result of mathematical expression evaluation from the computer cluster node, execute at least a second mathematical expression evaluation using the result of the first mathematical expression evaluation, and communicate a result of the second mathematical expression evaluation to the first node.” The parties’ dispute over the appropriate construction of this limitation is addressed by multiple claim terms at varying levels of specificity, as shown below:

Claim Term	ACS Construction	NVIDIA Construction
the first node	the <u>second</u> node	Indefinite, lacks antecedent basis.
communicate a result of the second mathematical expression evaluation to the first node	communicate a result of the second mathematical expression evaluation to the <u>second</u> node	Indefinite
wherein the third node is configured to receive the result of mathematical expression evaluation from the computer cluster node, execute at least a second mathematical expression evaluation using the result of the first mathematical expression evaluation, and communicate a result of the second mathematical expression evaluation to the first node	wherein the third node is configured to receive the result of mathematical expression evaluation from the computer cluster node, execute at least a second mathematical expression evaluation using the result of the first mathematical expression evaluation, and communicate a result of the second mathematical expression evaluation to the <u>second</u> node	Indefinite.

(emphasis added to highlight change.)

177. The relevant part of Claim 35 recites

A **computer cluster node** for evaluating expressions in parallel with other computer cluster nodes, the computer cluster node comprising:

...

program code that, when executed, is capable of causing the hardware processor to:

receive calls from **a second node** comprising a second hardware processor configured to access a second memory comprising program code for a user interface and program code for a second single-node kernel, the second single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution;

execute, using the hardware processor, at least a first mathematical expression evaluation; and

communicate a result of the first mathematical expression evaluation to **a third node** comprising a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the computer cluster node, execute at least a second mathematical expression evaluation using the result of the first mathematical expression evaluation, and communicate a result of the second mathematical expression evaluation to **the first node**;

wherein the user connection interface is configured to return at least one result of mathematical expression evaluation to a user interface or a script; and

...

(emphasis added.) A person of ordinary skill in the art would have understood that the recitation of “the first node” is an error, and that the claim should recite “the second node.”

a. “The first node” is a clear drafting error

178. Upon reading Claim 35, a person of ordinary skill in the art would immediately realize there is a clear error because “the first node” lacks antecedent

basis.

179. The prosecution history is consistent with my understanding that the error would have been clear to any person of ordinary skill in the art. NVIDIA Corp. filed a petition seeking *inter partes* review of Claim 35 of the '768 Patent. NVIDIA Corp. submitted a declaration from its expert, Dr. Henry Tufo, in support thereof. Dr. Tufo testified that his background and expertise qualified him as an expert in the technical issues of the '768 Patent. (IPR2021-00020 Tufo Declaration at 1-5.) Dr. Tufo analyzed Claim 35 and noted that “‘the first node’ lacks antecedent basis and likely is a drafting error.” (*Id.* at 80.) Dr. Tufo’s recognition of the error confirms my understanding that the error would be readily apparent to a person of ordinary skill in the art.

b. The correction is not subject to reasonable debate and is supported by the prosecution history

180. It would have been equally apparent to a person of ordinary skill in the art that the “the first node” should have referenced “the second node,” and that the drafting error was introduced into Claim 35 through an Examiner’s amendment, discussed further below. (November 29, 2018 Notice of Allowability at 11-12); November 29, 2018 Examiner-Initiated Interview Summary.)

181. A person of ordinary skill in the art reviewing Claims 1, 26, and 29 would have observed a common pattern within the recited movements of the results of the mathematical expression evaluations: a first node is configured to access

program code for a user interface; a second node is configured to receive calls from the first node, execute a first mathematical expression evaluation, and communicate the result of the first mathematical expression to a third node; and the third node is configured to receive the result of the first mathematical expression evaluation, execute a second mathematical expression evaluation, and communicate the result of the second mathematical expression evaluation to the first node, *i.e.*, the user interface node.

182. A person of ordinary skill in the art reviewing Claim 35 would have observed that the reference to “the first node” breaks the common pattern observed in Claims 1, 26, and 29. In Claim 35 it is a second node that is configured to access program code for a user interface. To maintain the common pattern, the claim should communicate the result of the second mathematical expression evaluation to the second node, *i.e.*, the user interface node.

183. The prosecution history confirms that “the first node” referenced in Claim 35 should be corrected to “the second node.” In a November 2018 interview, the Examiner proposed incorporating draft dependent Claim 26 into Claim 35¹³ and re-writing Claim 35 “to have all the limitations of th[e] other independent claims”

¹³ References to the prosecution of Claim 35 should be understood to refer to prosecution of draft Claim 37, which issued as Claim 35. (November 29, 2018 Issue Classification at 3.)

in order to put the application in condition for allowance. (November 29, 2018 Examiner-Initiated Interview Summary.) ACS's representative authorized the amendment. (*Id.*) The resulting Examiner amendment is reproduced below:

37. **(Currently amended)** A computer cluster node for evaluating expressions in parallel with other computer cluster nodes, the computer cluster node comprising:

a hardware processor configured to access one or more non-transitory memory devices comprising program code for a single-node kernel that, when executed, causes the hardware processor to interpret user instructions, to evaluate mathematical expressions, and to produce results of mathematical expression evaluation, wherein the hardware processor comprises multiple processor cores;

a user connection interface configured to receive a command to start a cluster initialization process for a computer cluster;

a mechanism to communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture; and

~~program code that, when executed, is capable of causing the hardware processor to receive calls from a first node, execute at least a first mathematical expression evaluation, and communicate a result of mathematical expression evaluation to another node for use in at least a second mathematical expression evaluation;~~

program code that, when executed, is capable of causing the hardware processor to:
receive calls from a second node comprising a second hardware processor
configured to access a second memory comprising program code for a user
interface and program code for a second single-node kernel, the second single-node
kernel configured to interpret user instructions and distribute calls to at least one of
a plurality of other nodes for execution;

execute, using the hardware processor, at least a first mathematical
expression evaluation; and

communicate a result of the first mathematical expression evaluation to a
third node comprising a third hardware processor with a plurality of processing
cores, wherein the third node is configured to receive the result of mathematical
expression evaluation from the computer cluster node, execute at least a second
mathematical expression evaluation using the result of the first mathematical
expression evaluation, and communicate a result of the second mathematical
expression evaluation to the first node;

wherein the user connection interface is configured to return at least one result of
mathematical expression evaluation to a user interface or a script; and

wherein the computer cluster node is configured to:

accept user instructions;

after accepting user instructions, communicate at least some of the user
instructions using the mechanism for the nodes to communicate with each other;
and

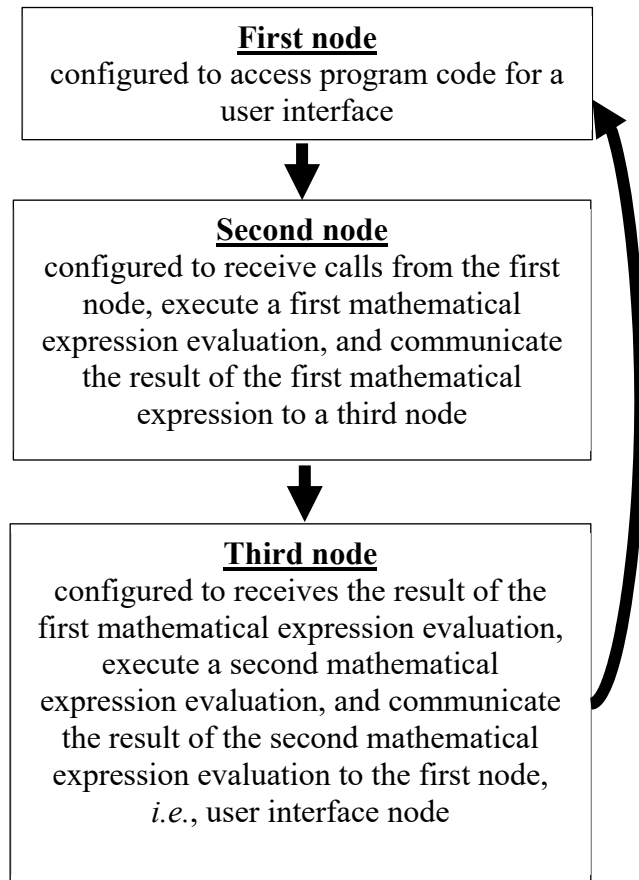
after communicating at least some of the user instructions using the
mechanism, communicate at least some of the user instructions to the single-node
kernel.

(November 29, 2018 Notice of Allowability at 11-12.) The Examiner added the orange highlighted language to incorporate the limitations on which draft Claim 26 depended. (*Id.*) The Examiner added the blue highlighted language to incorporate draft dependent claim 26 into Claim 35. (November 29, 2018 Examiner-Initiated

Interview Summary.) This Examiner amendment introduced the antecedent basis error.

184. The Examiner's error arose because each of the other independent claims were directed to "[a] computer cluster comprising: a plurality of nodes," *i.e.*, a cluster. (November 29, 2018 Notice of Allowability at 2.) Conversely, Claim 35 was directed to "[a] computer cluster node for evaluating expressions in parallel with other computer cluster nodes, the computer cluster node comprising," *i.e.*, a node. (*Id.*) Re-writing Claim 35 to have all the limitations of the other independent claims therefore required different wording to accommodate the different subject matter.

185. Knowing the Examiner's intention when amending Claim 35 makes the clerical correction clear. As noted above, the other independent claims share a common pattern of recited movements of the results of the mathematical expression evaluations, depicted below:



186. ACS described this communication pattern when the corresponding limitations were added to the independent claims as follows:

Applicant has amended the independent claims in an effort to advance prosecution of the application. No new matter is added. The application as filed explains that ***a first node of a computer cluster can be connected to a user interface*** or a script and configured to ***receive user instructions from the user interface*** or the script. *E.g.*, application as filed at ¶¶ [0012], [0025]. ***The first node can distribute calls to other cluster nodes.*** *E.g.*, *id.* at ¶ [0026]. The cluster nodes can evaluate expressions according to a set of rules. *E.g.*, *id.* at ¶ [0079]. ***The application describes an intercommunication architecture and functions that permit intermediate results to be passed from one node to another node without involvement of the first node.*** *E.g.*, *id.* at ¶¶ [0027], [0121]. For example, in a non-limiting embodiment, the

application describes a function that allows elements near the edge of a list, which can include intermediate results of evaluation, to be copied to neighboring processors so that neighboring edges of each node partition can interact. *Id.* at ¶¶ [0080], [0137]. ***Results of evaluations can be communicated back to the first node and returned to the user interface*** or script. *E.g., id.* at ¶¶ [0026], [0133].

(October 5, 2018 Response to Non-Final Office Action Dated April 13, 2018 at 13-14 (emphasis added).)

187. ACS's argument makes it clear that the intent of the limitation was to have an instruction begin at a first node, have corresponding data move through the other nodes for subsequent evaluation without involving the first node, and then ultimately return the result of the final evaluation to the first node so it could be returned to the user interface. (*See id.*)

188. ACS concluded by stating:

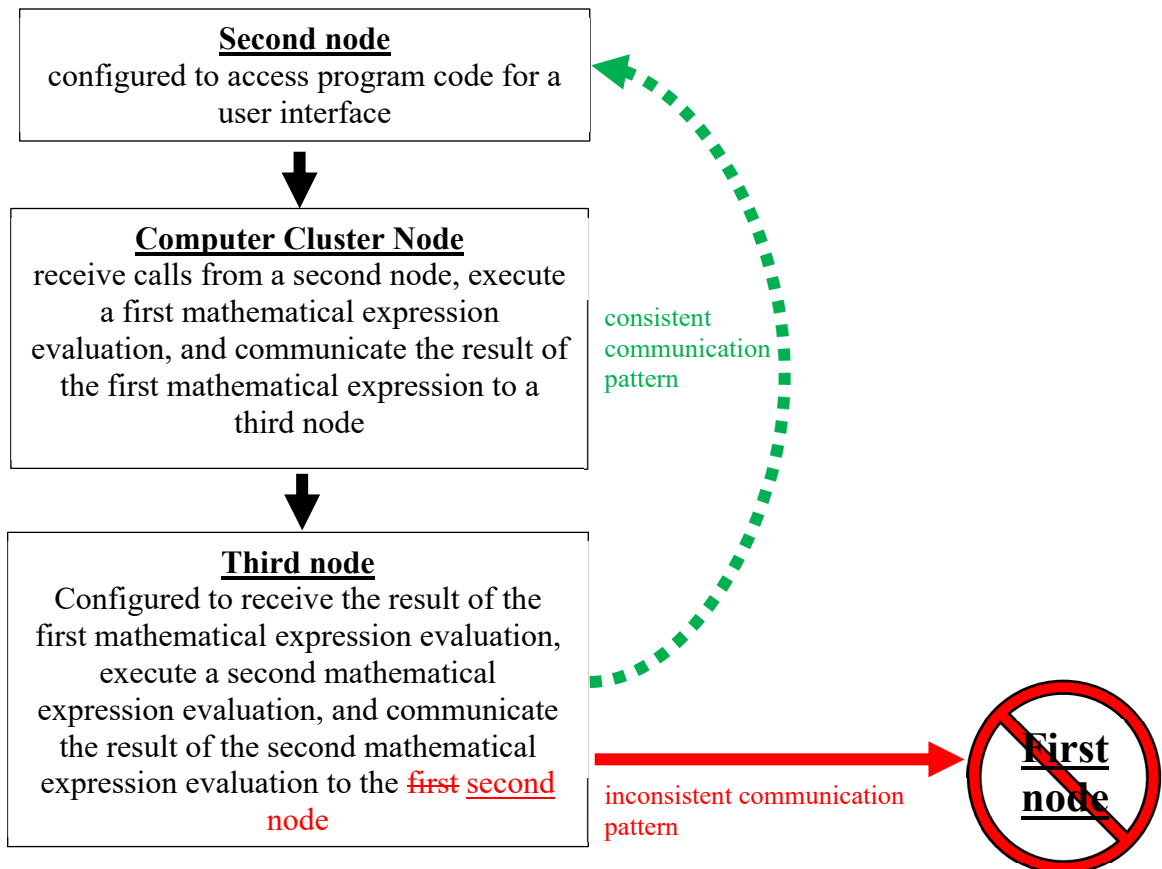
As explained in the interview of May 15, 2018, the cited references do not disclose the combination of features recited in the claims as amended. For example, the references even if combined do not disclose a first node configured to interpret user instructions and distribute calls to other nodes for execution; a second node configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result to a third node; and a third node configured to receive the result, execute at least a second mathematical expression evaluation using the result, and communicate the result of the second mathematical expression evaluation to the first node, in combination with the other features recited in Claim 2.

(*Id.* at 14.)

189. Following this, the Examiner proposed re-writing independent Claim 35 “to have all the limitations of th[e] other independent claims.” (November 29,

2018 Examiner-Initiated Interview Summary.)

190. In re-writing the limitation for use in Claim 35, the Examiner included the erroneous reference to “the first node” that breaks the common communication pattern from the other independent claims that the Examiner intended to include in Claim 35. To maintain the common communication pattern, the third node in Claim 35 would send the result of the second mathematical expression evaluation to the node connected to the user interface, which is the second node:



191. A tabular comparison of the relevant data movements from Claim 1 and Claim 35 is below. The nodes’ names, structures, and roles have been color-coded

to emphasize their correlation.

Claim 1	Claim 35	Description
A computer cluster comprising: . . .	A computer cluster node for evaluating expressions in parallel with other computer cluster nodes, the computer cluster node comprising: . . .	Claim 35 computer cluster node corresponds to Claim 1 second node.
a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and	program code that, when executed, is capable of causing the hardware processor to receive calls from a second node comprising a second hardware processor configured to access a second memory comprising program code for a user interface and program code for a second single-node kernel, the second single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution;	Claim 35 second node corresponds to Claim 1 first node.
a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of the first mathematical expression evaluation to a third node;	execute, using the hardware processor, at least a first mathematical expression evaluation; and communicate a result of the first mathematical expression evaluation to a third node	Claim 35 computer cluster node corresponds to Claim 1 second node.
wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to	comprising a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of	Claim 35 third node corresponds to Claim 1 third node. Based on the recited

receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;	mathematical expression evaluation from the computer cluster node, execute at least a second mathematical expression evaluation using the result of the first mathematical expression evaluation, and communicate a result of the second mathematical expression evaluation to the first node;	functionality and the pattern of movement of the results of mathematical expression evaluations, Claim 35 “first node” also corresponds to Claim 1 first node, and thus should read the “second node.”
---	--	--

192. Therefore, a person of ordinary skill in the art would have understood in view of the prosecution history that Claim 35 should be corrected to change “the first node” to “the second node.” No other correction would be consistent with the prosecution history. If “the first node” was interpreted to be the “computer cluster node” recited in the preamble, the second mathematical expression evaluation would not be returned to the user interface node. If “the first node” was interpreted to introduce a new node to the claim, *i.e.*, interpreted as “a first node,” there likewise would not be a return of the second mathematical expression evaluation to the user interface node, or a node previously receiving any part of the user instruction or performing any corresponding mathematical expression evaluation at all. Thus, a person of ordinary skill in the art would have understood that the only interpretation of “the first node” consistent with the specification and prosecution history is “the second node.”

IV. MATERIALS CONSIDERED

193. In addition to my knowledge, expertise, and decades of relevant experience in the relevant field, I considered the following materials when preparing this declaration.

- a) The '768 Patent;
- b) The prosecution history of the '768 Patent;
- c) The IPRs of the '768 Patent and the Related Patents, including the filed declarations;
- d) U.S. Patent 8,082,289;
- e) The extrinsic evidence identified by Defendants;
- f) All other documents referenced above.

194. I reserve the right to consider any additional materials that are brought to my attention.

V. COMPENSATION

195. I am being compensated for my time at my standard rate of \$800 per hour for my work on this litigation. My compensation is in no way contingent on the outcome of this litigation.

I. RESERVATION OF RIGHTS TO SUPPLEMENT OR MODIFY

1. I reserve the right to supplement or modify my opinions in the event that additional information comes to my attention.

2. In particular, I reserve the right to supplement my opinions to address the construction of disputed claim limitations in light of any opinions offered by Defendants' expert, who I have been informed is Ian Foster, regarding the construction of disputed claim limitations.

3. Moreover, if called to testify before the Court, I may testify about matters: (1) raised on direct or cross examination; (2) necessary to rebut any matters that the Court allows Defendants to introduce or rely upon; or (3) otherwise raised before the Court in relation to matters set forth in this report and any other supplemental reports I submit. In addition to the items identified above, my testimony before the Court may also be based, in part, upon the testimony of fact witnesses or other expert witnesses.

I declare under penalty of perjury that the foregoing is true and correct.



Dated: October 27, 2021

Jaswinder Pal Singh, Ph.D.

CERTIFICATE OF SERVICE

I hereby certify that on October 27, 2021, I served the foregoing document via e-mail on Defendant's counsel of record as follows:

nvidia-acs-dla@dlapiper.com

Brian A. Biggs (DE Bar No. 5591)
Erin E. Larson (DE Bar No. 6616)
DLA PIPER LLP (US)
1201 North Market Street, Suite 2100
Wilmington, DE 19801-1147
brian.biggs@dlapiper.com
erin.larson@dlapiper.com

Mark Fowler (*Pro Hac Vice*)
Clayton Thompson (*Pro Hac Vice*)
Jake Zolotorev (*Pro Hac Vice*)
Carrie Williamson (*Pro Hac Vice*)
Asa Wynn-Grant (*Pro Hac Vice*)
Monica De Lazzari (*Pro Hac Vice*)
2000 University Avenue
East Palo Alto, CA 94303-2214
mark.fowler@dlapiper.com
clayton.thompson@dlapiper.com
jake.zolotorev@dlapiper.com
carrie.williamson@dlapiper.com
asa.wynn-grant@dlapiper.com
monica.delazzari@dlapiper.com

Dated: October 27, 2021

/s/ Cheryl T. Burgess

Cheryl T. Burgess

EXHIBIT A

Jaswinder Pal Singh

Professor, Computer Science Department
Princeton University, Princeton, NJ 08544
(609) 258-5329, jpgs@cs.princeton.edu

EDUCATION

Stanford University. Ph.D. in Electrical Engineering, 1993. Advised by Prof. John Hennessy and Prof. Anoop Gupta. My thesis research was at the boundary of applications and parallel computer systems, including programming models, software and hardware.

Stanford University. M.S. in Electrical Engineering, 1989.

Princeton University. B.S. in Electrical Engineering and Computer Science, 1987, *summa cum laude*.

PROFESSIONAL EXPERIENCE

- 7/05 – present Professor, Computer Science Department, Princeton University.
- 9/19 – present Director of Undergraduate Studies (Majors), Computer Science Department, Princeton University.
- 10/13 – present Member of the Board of Directors, 8x8 Inc. (NASDAQ:EGHT).
- 1/19 – present Member of the Board of Directors, Blockstack, Inc. A private company in blockchain technology.
- 9/14 – present Member of the Board of Trustees, The Dalton School, New York.
- 7/14 – present Director, Program in Applications of Computing, Princeton University. A program for about 125 non-Computer Science majors.
- 9/99 – 2010 Director, Program in Integrative Computer and Application Sciences (PICASso), a multi-department, university-wide interdisciplinary program focused on scalable parallel and distributed computing at the boundary of computer science and a broad range of application sciences, . Developing and fostering new interdisciplinary courses, new practice-oriented workshops in scalable computing based on needs gathered from multiple departments, new interdisciplinary seminar series, and programs in effective written and oral scientific communication, as well as funding students and running the program. Creating a core of courses for an interdisciplinary computational science curriculum, taught by faculty in different departments, in addition to parallel and distributed computing courses.
- 7/03 – present Executive Committee, Princeton Institute for Computational Science and Engineering (PICSsiE). A permanent institute that came out of the PICASso program that I led, and that has taken over its activities as well, including a permanent Graduate Certificate Program in Computational Science and Engineering at Princeton that I developed.
- 9/05 – 9/07 Director of Princeton portion of Center for Modeling Viral Immunity and Antagonism, a \$20+M NIH Center involving Mount Sinai School of Medicine and Princeton University. Member, Executive Committee of Center.
- 9/00 – 6/05 Co-founder and Chief Technical Officer, firstRain, Inc. (on leave from Princeton University 9/00 to 6/02). Developing novel Internet/Web applications and Internet infrastructure. Improved and developed new protocols for web-based communication and information access.
- 7/99 – 6/05 Associate Professor, Computer Science Department, Princeton University.
- 3/99 – present Member of the Technical Advisory Board, NOAA Geophysical Fluid Dynamics Laboratory. Aiding in the shift from vector to parallel computing, and in the procurement of large parallel computing and analysis infrastructures.
- 2/95 – 6/99 Assistant Professor, Computer Science Department, Princeton University. Performing research in parallel computer systems and applications. Focus on (i) scalable multiprocessing in both the shared address space and message passing paradigms, including studying the programming and performance tradeoffs between the two, (ii) understanding how shared address space abstractions might be supported using commodity parts, and how applications might be restructured for such systems, and

(iii) the use of high-performance computing in biology, medicine, cosmology and computer graphics. Advising several Ph.D. students in these areas.

- 2/93 - 1/95 Research Associate, Stanford University. Leading the parallel applications effort in the Stanford DASH/FLASH research groups, and studying the implications of these applications for the design of multiprocessor software and hardware systems. Evaluating performance of real and simulated systems, and developing methodologies for the above. Expanding the SPLASH suite of parallel applications to SPLASH-2. Advising several Ph.D. students.
- 10/87 – 2/93 Research Assistant, Stanford University. Developing parallel scientific and engineering applications and studying their implications for parallel processing systems, software and architecture. Studying the scaling of applications and architectures. Creating and distributing the SPLASH suite of parallel applications.

HONORS

- *Presidential Early Career Award for Scientists and Engineers (PECASE)*, 1997. Awarded to twenty young scientists and engineers in the United States selected from all areas of science/engineering by the National Science Foundation.
- *Sloan Research Fellowship*, 1997. Awarded to about 10 young computer scientists every year.
- Paper selected as DEBS 2005 Most Influential Paper. May 2019.
- Paper selected as Best student paper in USENIX USITS Symposium, 2000.
- Paper selected for journal publication from the International Conference on Supercomputing, 1999.
- Paper selected for Journal of Computational Biology from the RECOMB computational biology conference, 1998.
- Paper selected for journal publication from the Symposium on Parallel Algorithms and Architectures, 1996.
- Paper selected for journal publication from the Symposium on Parallel Algorithms and Architectures, 1996.
- Paper nominated for Best Student Paper Award for Supercomputing'97.
- Two papers selected among best 5 submitted to the Intl. Symposium on Computer Architecture (1992 and 1993), one of them as "Impact Paper" for the Federated Computer Research Conferences, 1993.
- *Summa cum laude*, Princeton University. Princeton University Undergraduate Scholarship.
- Member of Phi Beta Kappa, Tau Beta Pi and Sigma Xi since 1987.

TEACHING

Have taught core undergraduate courses as well as graduate-level courses in Parallel Computing, Parallel Architecture and Programming, Scalable Internet services, applications, protocols and infrastructure, and in analysis of data. Created and taught several new interdisciplinary courses I created for the PICASSo program, catering to and attracting students from many different departments on campus, from Sciences, Engineering, Operations Research and Computer Science. Created and taught courses in Marketplace Design and at the boundary of technology, business, marketplaces and economics. Created and taught a new type of course for Princeton called Innovating at the Boundary of Technology, Business and Marketplaces (dubbed the "CTO Course").

Ph.D. DISSERTATIONS SUPERVISED

- Adrian Soviani. New Programming Models for Scalable Parallel Computing.
- Erich Schmidt. Query-independent ranking for large-scale publish-subscribe systems.
- Chi Zhang. A Peer-to-peer substrate for Distributed Search Trees.
- Fengyun Cao. MEDYM: A distributed content-based publish-subscribe system.
- Mao Chen. Using Explicit User Information to Improve Internet Services.
- Yefim Shuf. Improving the Memory Performance of Java Programs.
- Steven Kleinstein. Towards Quantitative Modeling of Immune System Dynamics.
- Hongzhang Shan. A Comparison of Programming Models for High-Performance Parallel Computing.
- Dongming Jiang. Performance Portability on Shared Address Space Architectures.
- Angelos Bilas, Using Network Interfaces to Accelerate Software Shared Memory.
- Liviu Iftode, Shared Virtual Memory using Automatic Update. With Kai Li.

- Steven Cameron Woo, Stanford University. Integrating Block Data Transfer in Cache-coherent Multiprocessors. With John Hennessy.
- Chris Holt, Stanford University. Application and Architectural Bottlenecks in Distributed Shared Memory Multiprocessors, expected 2003. With John Hennessy.
- Cheng Che Chen. Protein Structure Determination in the Presence of Uncertainty: Algorithms and Parallelism.
- Alexander Kozlov, Stanford University. Probabilistic Inference in Belief Networks: Algorithms and Parallelism. With John Hennessy and later Daphne Koller.

Served on dissertation committees of several others.

PROGRAM COMMITTEES

Have served on the program committees and as program chair and track chair of many ACM, IEEE and other conferences in scalable computing, including the International Symposium on Computer Architecture, Supercomputing/SC, SIGMETRICS Conf. on Measurement and Modeling of Computer Systems, Principles and Practice of Parallel Programming, Symposium on Parallel Algorithms and Architectures, International Conference on Supercomputing, International Parallel Processing Symposium, International Conference on Parallel Processing etc., as well as several ACM and IEEE workshops on scalable computing, data analysis and mining, architecture, and languages and compilers.

INVITED EVALUATION COMMITTEES AND TALKS

Have served on several invited government research/education program and proposal evaluation committees, including for the US Government (NOAA's procurement programs and the National Science Foundation) and the Swedish Government (Evaluating the ARTES/PAMP national program for scientific research and graduate education in high-performance computing, as well as other programs for the Swedish Research Council). Have also served as an external evaluator for faculty hiring at Uppsala University, Sweden and the University of Copenhagen, Denmark.

Have delivered many invited presentations and lectures at a variety of international and domestic venues, including:

- Conference panels and invited presentations at ISCA, ASPLOS, SIAM and other conferences and workshops
- NSF and DARPA PI meetings in scalable computing and interdisciplinary research and education
- International government programs (Sweden and India, including India's cross-IIT TECHFEST)
- U.S. universities such as UC Berkeley, U. Washington, Stanford, NYU, U of Toronto, SUNY, etc.
- International universities and research laboratories/centers in France, India, Italy, Japan, Sweden, and South Africa.
- Industrial research laboratories such as Microsoft, Sun, IBM, Panasonic, Siemens etc.
- Government research laboratories such as NASA, Sandia, Argonne, PPPL (plasma physics) and GFDL (climate)
- Industrial research and product/business organizations; e.g. Yahoo, Microsoft, Endeca, etc.
- Several industrial conferences and trade shows, industrial/academic and executive summits involving web-based information access, search, dissemination/communication and intelligence.

PATENTS

Patents awarded:

Method and apparatus for focused crawling. United States Patent 20060277175. Pub. Date 12/07/2006.

Method and apparatus for focused crawling. United States Patent 7080073. Pub. Date 07/18/2006.

Method and apparatus for searching network resources. United States Patent 6915294. Pub. date 07/05/2005.

Method and apparatus for searching network resources. United States Patent 20050210018. Pub. date 09/22/2005

Method and apparatus for searching network resources. United States Patent 7415469. Pub. date 09/05/2006.

Method and apparatus for extracting relevant network data. United States Patent 7103838. Publication date 09/05/06.

SAMPLE GRANTS RECEIVED

(Grants in which I am Principal Investigator, not Co-Principal Investigator. Pending grant proposals not included.)

Princeton CTO Research and Education, \$800,000, 2010. Additional funds being secured.

National Institutes of Health. HHSN266200500021. \$20,250,000. PI of Princeton Portion.

National Science Foundation IGERT Grant, No. DGE-9972930. Funds up to fourteen graduate students per year in scalable internet services, computing and data analysis. \$2,172,000.

Accessible Climate Computing for 'Downstream' Science: An Energy and Climate Grand Challenge Proposal. Princeton Grand Challenges Program. \$100,000 per year.

National Science Foundation. Presidential Early Career Award for Scientists and Engineers (PECASE). Only computer scientist to receive it that year. \$500,000.

National Science Foundation. CISE Research Award. \$980,000.

National Science Foundation. CISE Instrumentation Award. Three projects in applications and scalable computing. \$2,172,000.

National Science Foundation. Adaptive Performance Portable Software for Next-Generation Applications. \$600,000.

firstRain, Inc. Collaborative Student Research Grant.

National Oceanic and Atmospheric Administration Grant.

Alfred P. Sloan Foundation. Sloan Research Fellowship. \$35,000.

PROFESSIONAL ORGANIZATIONS

The Association for Computing Machinery

Institute of Electrical and Electronics Engineers, Inc.

Phi Beta Kappa, Tau Beta Pi, Sigma Xi

PUBLICATIONS

One leading graduate/practitioner level textbook (below) and over 100 refereed conference and journal publications (available upon request).

Textbook

David E. Culler, Jaswinder Pal Singh. *Parallel Computer Architecture: A Hardware-Software Approach*. With Anoop Gupta. Morgan Kaufmann Publishers, 1998. Leading textbook in parallel architecture and programming.

EXHIBIT E

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

ADVANCED CLUSTER SYSTEMS, INC.

Plaintiff,

v.

NVIDIA CORPORATION, NVIDIA
SINGAPORE PTE. LTD., and NVIDIA
INTERNATIONAL, INC.

Defendants.

CASE NO. 19-2032-CFC-CJB

DECLARATION OF DR. IAN FOSTER

I. INTRODUCTION

1. My name is Dr. Ian Foster and I am over 18 years of age and competent to make this declaration. If called to testify as a witness in this matter, I could and would testify truthfully to each of the matters set forth below.

2. I am the Arthur Holly Compton Distinguished Service Professor of Computer Science at the University of Chicago and a Distinguished Fellow, Senior Scientist, and Division Director at Argonne National Laboratory.

3. I have been retained to serve as an independent technical expert in this matter by Defendants NVIDIA Corporation, NVIDIA Singapore PTE LTD., and NVIDIA International, Inc. (“NVIDIA”).

4. I am being compensated for my work on this case at my standard consulting rate of \$600 per hour. I am being reimbursed for any out-of-pocket expenses incurred in relation to my work. My compensation does not depend on the outcome of this litigation or on the nature of any opinions I provide. I am not aware of having any other interests in this case or in any of the parties.

5. I have been asked by counsel for NVIDIA to provide my opinions with respect to how a person of ordinary skill in the art (“POSITA”) would have understood certain words or phrases that appear in the claims of U.S. Patent No. 10,333,768 (“’768 patent”).

II. MATERIALS CONSIDERED

6. In preparing this declaration, I have reviewed the following documents:

- The ’768 patent and its file history.
- The parties’ identification of preliminary constructions, and evidence identified by the parties in their respective claim construction disclosures.
- Webster’s New World Dictionary of Computer Terms (6th ed. 1997), at 280 (defining “kernel”).
- The American Heritage College Dictionary (3rd ed. 1993), at 711 (defining “interpreter”).
- Excerpts from Mathematica 5.0 Reference Guide.

- Excerpts from MAPLE V Learning Guide, Release 5.
- IEEE 100, The Authoritative Dictionary of IEEE Standards Terms (7th ed. 2000), at 804 (defining “peer-to-peer communication”).
- U.S. Patent No. 9,141,355.
- U.S. Patent No. 7,747,981.
- Dorrigiv et al., “Search Algorithms for Unstructured Peer-to-Peer Networks” (2007).
- Tewari et al., “Optimal Search Performance in Unstructured Peer-to-Peer Networks With Clustered Demands” (2007).
- Gkantsidis et al., “Hybrid Search Schemes for Unstructured Peer-to-Peer Networks” (2005).
- Van Roy, “Overcoming Software Fragility with Interacting Feedback Loops and Reversible Phase Transitions” (2008).
- Documents relating to IPR2021-00019, including the Petition for *Inter Partes* Review, Patent Owner’s Preliminary Response, Petitioner’s Reply to Preliminary Response, Patent Owner’s Consolidated Sur-Reply in Support of its Preliminary Response, and Decision Denying Institution.
- Documents relating to IPR2021-00020, including the Petition for *Inter Partes* Review, Patent Owner’s Preliminary Response, Petitioner’s Reply to Preliminary Response, Patent Owner’s Consolidated Sur-Reply in Support of its Preliminary Response, and Decision Denying Institution.

7. In reaching my opinions, I relied on my background, knowledge, and experience in distributed computing.

III. BACKGROUND AND QUALIFICATIONS

8. A true and correct copy of my *curriculum vitae* is included as Exhibit A to this declaration.

1 **A. Educational Background**

2 9. I hold a Bachelor of Science (B.Sc. Hons I) degree in Computer Science from the
3 University of Canterbury (Christchurch, New Zealand), and a Doctor of Philosophy (Ph.D.)
4 degree in Computer Science and Diploma of Imperial College from Imperial College (London,
5 United Kingdom).

6 10. My work as a computer scientist has been at the intersection of computing and the
7 sciences. My work has produced both practical technologies that have seen wide adoption and
8 concepts and methods that have proven influential in research and education.

9 11. My research interests span a range of topics in parallel, distributed, and data-
10 intensive computing. A unifying theme of these topics is a desire to use the power of rapid
11 communication to accelerate discovery, whether by linking people with remote computers and
12 data, accelerating complex computational processes, or enabling distributed virtual teams. I often
13 build sophisticated artifacts (software and distributed systems) in order to apply, evaluate, and
14 disseminate new concepts and methods.

15 12. A major focus of my research since the mid-1990s has been the methods and
16 technologies for resource sharing in networked environments. In 1995, I led software research
17 and development for the I-WAY wide-area distributed computing experiment, which connected
18 supercomputers, databases, and other specialized resources at 17 sites across North America. In
19 1996, I founded the Distributed Systems Laboratory at Argonne National Laboratory and the
20 multi-institutional Globus project at the University of Chicago, which developed core
21 technologies for resource discovery, scheduling, configuration, security, data access, and
22 execution in high-performance networked environments. The resulting technologies formed the
23 basis for many national and international “Grid” projects funded by DOE, NASA, NSF, the
24 European Union, and the UK eScience Program, in addition to influencing grid computing
25 products worldwide.

26 13. With the emergence of large-scale commercial clouds (“grids with a business
27 model”) in the 2000s, and the growing importance of large digital data in science, my research in
28

1 distributed computing has increasingly focused on methods for outsourcing and automating
2 research data management tasks. This effort has led to the development of cloud-based Globus
3 services, which are seeing broad adoption both in the U.S. and internationally.

4 14. I have served as Principal Investigator (PI) or Co-PI on projects connected to the
5 National Computational Science Alliance; the National Science Foundation (NSF) Community
6 Development of Globus Software (CDIGS) project; the Department of Energy (DOE) Center for
7 Enabling Distributed Petascale Science (CEDPS); the DOE/NSF Open Science Grid and the NSF
8 TeraGrid; and other DOE and NSF programs. I am currently PI for the DOE Codesign Center for
9 Online Data Analysis and Reduction (CODAR).

10 **B. Publications and Patents**

11 15. I have published eight books and over 600 articles. My publications have been
12 cited in other scientific papers more than 130,000 times.

13 **C. Honors and Awards**

14 16. I am an elected Fellow of the American Association for the Advancement for
15 Science, the Association for Computing Machinery, the British Computer Society, and the
16 Institute of Electrical and Electronic Engineers. I have been awarded the Lovelace Medal of the
17 British Computer Society, the IEEE-CS Goode Award, the IEEE-CS Charles Babbage Award, the
18 IEEE Tsutomu Kanai Award, the IEEE TCSC Award for Excellence in Scalable Computing, and
19 the inaugural ACM HPDC Lifetime Achievement Award. And I have received honorary
20 doctorates from the University of Canterbury, New Zealand, and CINVESTAV, Mexico.

21 **IV. LEGAL PRINCIPLES**

22 17. NVIDIA's counsel has informed me of and asked me to apply the following legal
23 principles in developing my opinions that I express in this declaration.

24 18. I understand that, in general, the terms of a patent claim are given the meaning
25 they would have had to a person of ordinary skill in the art at the time the invention was made, in
26 view of the intrinsic evidence, which includes the patent's claims, specification, and prosecution
27 history. I further understand that a patent owner can "act as his own lexicographer," which I
28

1 understand to mean that a patent can define a term to have a meaning other than the one that a
2 person of skill in the field would normally give it. I further understand that a patent owner can
3 define his own phrases or terms, and that the meaning of those phrases or terms would then come
4 from the patent. I have been informed that such a definition may be express or implied.

5 19. I understand that in assessing how a person of ordinary skill in the art would
6 understand a claim term, it is permissible, and sometimes helpful, to consult other evidence that is
7 not part of intrinsic evidence, such as testimony, dictionary definitions, and technical treatises,
8 commonly referred to as “extrinsic evidence.”

9 20. I understand that a person of ordinary skill in the art relevant to a patent is a
10 hypothetical person who is presumed to have known all of the relevant prior art as of the priority
11 date of the patent. I further understand that factors that may be considered in determining the
12 level of ordinary skill in the art may include: (a) the educational level of the inventor; (b) the type
13 of problems encountered in the art; (c) the prior art solutions to those problems; (d) the rapidity
14 with which innovations are made; (e) the sophistication of the technology; and (f) the educational
15 level of active workers in the field.

16 21. I was asked to provide my opinions as to the meaning of the disputed terms
17 “kernel,” “single-node kernel,” “the first node” recited in claim 35 and the claim limitation in
18 which the term appears, “peer-to-peer architecture,” the “mechanism” terms recited in claims 1,
19 26, 29, and 35, and “cluster node module.”

20 22. The opinions expressed in this declaration are based on my background, training,
21 experience, education, general knowledge of distributed computing, and my review of materials
22 relevant to this case.

23 23. I reserve the right to supplement or amend my opinions as permitted by any
24 applicable rules or by the Court if additional information is brought to my attention or in response
25 to submissions by the plaintiff in this case.

26 **V. SUMMARY OF THE '768 PATENT**

27 24. U.S. Patent No. 10,333,768 (“the ’768 patent”) was filed on May 14, 2014, and
28 claims June 13, 2006, as its earliest possible priority date.

1 25. The '768 patent is directed to "[a] computer cluster system comprising a plurality
2 of nodes." '768, Abstract. The '768 specification asserts that existing prior art "[g]rid computers
3 include a plurality of nodes" but the nodes "generally do not communicate with one another as
4 peers." '768, col 1:43-62, 6:32-38 ("Grid computing does not provide for communication
5 between slave nodes."). In the '768 patent, the term "node" generally refers to a "processing unit
6 or subunit that is capable of single-threaded execution of code." '768, col. 4:45-47. The node can
7 be a single microprocessor or a computer, some number of which can be arranged into a cluster.

8 26. One of the claim terms in dispute in this case is "kernel." The '768 patent asserts
9 that certain computer programs comprising "kernel" modules can benefit from running on a
10 cluster of nodes with communication between peer nodes. The specification discloses that kernel
11 modules can be conventional Mathematica or Maple kernels, for example. '768, col. 1:29-62,
12 4:14-23. Programs such as Mathematica and Maple are well-known in the art and solve
13 mathematical expressions or problems on a computer. They allow a user to input a mathematical
14 expression in the form of high-level computer commands (e.g., the Mathematica or Maple
15 commands) that is easily understood by a user even if they are not a professional computer
16 programmer. As the '768 patent specification explains, these high-level instructions representing
17 mathematical expressions are then *interpreted* by the kernel module's interpreter functionality
18 into low-level machine code that can be executed by the computer hardware to solve the
19 mathematical problem. '768, col. 23:9-12.

20 27. To provide the node intercommunication allegedly missing from the grid-
21 computing prior art, the '768 patent associates a kernel module on each node with a "cluster node
22 module." '768, col. 6:26-30. As shown in Fig. 2 below, each cluster node module communicates
23 with its associated kernel and with other cluster node modules. '768, col. 2:24-29, 5:33-55:
24
25
26
27
28

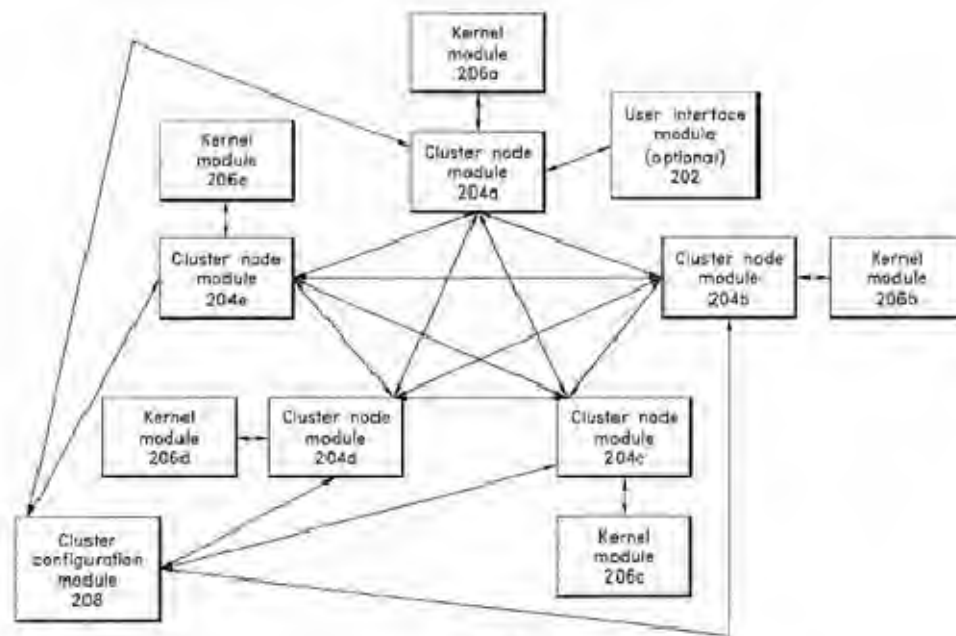


FIG. 2

28. The cluster node modules implement communications between the cluster nodes using the MPI (Message-Passing Interface) protocol. '768, col. 5:61-64, 6:10-20. The MPI protocol was conventional and well-known in the art as a tool enabling cluster nodes to communicate with each other. The conventional nature of MPI is acknowledged in the '768 patent specification through its incorporation by reference of the second provisional application that the inventors filed with the Patent Office, the '908 provisional. '768, col. 1:6-10.

29. The '908 provisional highlights the limited alleged advance of the alleged invention over the prior art by expressly recognizing that “[p]eer-to-peer” communication already was a standard “[c]ommunication pattern” for parallel cluster computing. '908 provisional, at 1, 4. Indeed, the '908 provisional recognizes that “[i]n the 1980s,” scientists found “that communication between different computing processors was *required*” for “the largest, most complex problems” in parallel computing. *Id.* (emphasis added). To provide this required intercommunication, “[i]n 1994, the Message-Passing Interface (MPI) standard was established.” *Id.*

1 **VI. OPINIONS PRESENTED**

2 **A. *Level of Ordinary Skill in the Art***

3 30. I have been informed by counsel understand that NVIDIA previously filed
4 petitions for *inter partes* review of the '768 patent and that NVIDIA's expert for those
5 proceedings, Dr. Henry Tufo, adopted the following standard of the level of ordinary skill in the
6 art: "A POSITA as of the filing date of the '768 Patent would have had a Bachelor's degree in
7 computer science, electrical engineering, or an equivalent field, and two years of academic or
8 industry experience working with distributed computing."

9 31. In my opinion, this is a reasonable standard for a POSITA as to the '768 patent and
10 I adopt this standard for this declaration. I am a person of ordinary skill in the art under this
11 standard.

12 **B. *"kernel" (Claims 1, 26, 29, 35)***

13 32. The term "kernel" appears in all of the asserted independent '768 patent claims:
14 claims 1, 26, 29, and 35. The term appears as part of the phrase "single node kernel." I address
15 the "single node" aspect separately below.

16 33. In my opinion, for the reasons I discuss below, the term "kernel," as it is used in
17 the '768 patent claims, is understood by a POSITA (a person of ordinary skill in the art) to mean:
18 *"program code for interpreting high-level code, commands, and/or instructions supplied by a*
19 *user or a script into low-level code, such as, for example, machine language or assembly*
20 *language."*

21 34. The term "kernel" did not at the time of the patent have a single accepted meaning
22 in the computing art. Instead, its meaning was (and is) context-specific—"kernel" can represent
23 different distinct concepts in computer science depending on the specific usage.

24 35. The term "kernel" sometimes refers to software implementing certain core
25 functions of an operating system, such as core functions in the well-known Windows, Linux or
26 Mac OS operating systems. Webster's Dictionary of Computer Terms provides a definition of this
27 type of "kernel" as: "In an operating system, the core portions of the program that reside in
28 memory and perform the most essential operating system tasks such as handling disk input and

output operations and managing the internal memory. Webster's New World Dictionary of Computer Terms, 6th Ed., 1997 at 280.

36. Another use of the term "kernel" is as an interpreter that converts high-level code (e.g., code entered by a user as user instructions) into low-level code that the machine can use. Discussions of software like Mathematica and Maple include examples of this use of "kernel." For example, the Mathematica 5.0 Reference Guide, published by Wolfram Research—the creators of Mathematica— describes the Mathematica "kernel" as the software that "actually performs computations." Mathematica 5.0 Reference Guide, section 1.0. It also describes how Mathematica performs computations: "Mathematica is *fundamentally an interpreter* which scans through expressions." *Id.*, section A.9.3 (emphasis added). An example of a high-level expression ("Integrate[x^n, x]") interpreted by Mathematica into low-level code that the computer can use is provided below:

Here is the integral $\int x^n dx$ in *Mathematica*.

In[1] := Integrate[x^n, x]

Out[1] = $\frac{x^{1+n}}{1+n}$

S. Wolfram, *The Mathematica Book*, 5th Ed., section 1.5.3.

37. The Maple V Learning Guide likewise describes the use of a "kernel" to refer to an interpreter: "The *kernel* is the base of Maple's system. It contains fundamental and primitive commands: the Maple language *interpreter* (which converts the commands you type into machine instructions your computer processor can understand), algorithms for numerical calculation, and routines to display results and perform other input and output operations." MAPLE V Learning Guide, Release 5, section 3.7 (emphasis added).

38. As described below, a POSITA would not understand the kernel that is claimed in

the '768 patent to be an operating system kernel. Instead, as I show below, the claimed "kernel" refers to a computer program that evaluates mathematical expressions by interpreting high-level code into the low-level code required for the computer itself.

Claim Language.

39. For convenience, I have reproduced '768 patent claim 1, highlighting where these terms appear:

1. A computer cluster comprising:

a plurality of nodes, wherein each of the plurality of nodes comprises a hardware processor, wherein one or more of the nodes are configured to receive a command to start a cluster initialization process for the computer cluster, and wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for *a single-node kernel* that, when executed, is capable of causing the hardware processor to evaluate mathematical expressions; and

a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture; wherein the plurality of nodes comprises:

a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for *a first single-node kernel, the first single-node kernel* configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and

a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of the first mathematical expression evaluation to a third node;

wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;

wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface;

wherein one or more of the nodes are configured to:

accept user instructions;

after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and

1 after communicating at least some of the user instructions using the
 2 mechanism, communicate at least some of the user instructions to *one or
 more single-node kernels*.

3 40. Several observations regarding the claimed “kernel” are evident to a POSITA from
 4 the claim language itself:

5 41. First, each of the claimed nodes is configured to access memory with “program
 6 code for a single-node kernel.” ’768, col. 30:28-30. Accordingly, the claimed kernel is software,
 7 or program code.

8 42. Second, the claimed kernel performs a specific claimed function: “when executed,
 9 is capable of causing the hardware processor to evaluate mathematical expressions.” ’768, col.
 10 30:31-32. Accordingly, the claimed kernel is not generic software. Rather, it is program code that
 11 evaluates mathematical expressions.

12 43. Third, the claim recites “a single-node kernel” on the “first node” that is
 13 “configured to interpret user instructions and distribute calls to at least one of a plurality of other
 14 nodes for execution.” ’768, col. 30: 40-43. “[U]ser instructions” refer to one type of high-level
 15 code that can be interpreted by an interpreter.

16 44. Fourth, at least some of these user instructions are communicated “to one or more
 17 single-node kernels.” ’768, col. 30:64-31:4.

18 45. These observations all show that the claimed “kernel” is not an operating system
 19 kernel but is instead an interpreter-type kernel that interprets high-level code into low-level code
 20 that a machine could use.

21 **Written Description.**

22 46. The rest of the specification provides a POSITA with further teaching regarding
 23 the meaning of the claimed “kernel.”

24 47. Starting with the Abstract: “In some embodiments, a computer cluster system
 25 comprises a plurality of nodes and a software package comprising a user interface and a *kernel*
 26 *for interpreting program code instructions*.” ’768, Abstract (emphasis added).

27 48. Thus, like the claim language discussed above, this description informs a POSITA
 28 that the claimed kernel is software that interprets program code instructions.

1 49. The boilerplate language “in some embodiments” does not detract from this
2 understanding for at least two reasons. First, as I show above, the claim language itself informs a
3 POSITA that the kernel is program code “configured to interpret user instructions.” As I explain
4 below, these “user instructions” are for evaluation of mathematical expressions, which is the
5 claimed function of the “kernel.” ’768, col. 30:31-32 (“[P]rogram code that, when executed, is
6 capable of causing the hardware processor to evaluate mathematical expressions.”).

7 50. Second, despite saying “in some embodiments,” the only embodiments disclosed
8 in the patent interpret high-level code into low-level code for evaluation of mathematical
9 expressions. Therefore, even where the specification prefaces its disclosure with a phrase such as
10 “in some embodiments,” as is the case with the Abstract, a POSITA reading the specification is
11 taught a single meaning of the claimed kernel that is consistent across the specification and the
12 claims.

13 51. Next, the Description of Related Art section, which addresses the alleged
14 invention’s background and applicability generally, explains that “[s]ome application programs
15 include an *interpreter* that executes instructions provided to the program by a user, a script, or
16 another source. *Such an interpreter is sometimes called a ‘kernel’* because, for example, the
17 interpreter can manage at least some hardware resources of a computer system and/or can manage
18 communications between those resources and software (for example, the provided instructions,
19 which can include a high-level programming language).” ’768, col. 1:29-37. After identifying the
20 term “kernel” in this manner, the specification then identifies Mathematica as a conventional
21 exemplar of such a kernel. *Id.* at 1:37-43.

22 52. Thus, this citation describes the term “kernel” as being used as an example of an
23 “*interpreter*” program that provides an interface between high-level code, such as “a high-level
24 programming language,” and the low-level “hardware resources.” *Id.* at 1:29-37. This is
25 consistent with the claims’ recitation of a kernel as interpreting user instructions. The
26 specification also suggests that this interpreter program is called a “kernel” by analogy to an
27 operating-system kernel, which can also manage hardware resources. But the teaching makes
28 clear that the “kernel” described in the specification is not itself an operating-system kernel.

1 53. Next, the specification devotes a separate section to the description of the “Kernel
2 Module” software. ’768, col. 23:8-26. There, it states: “***A kernel module 206 typically includes***
3 ***program code for interpreting high-level code, commands, and/or instructions supplied by a***
4 ***user or a script into low-level code***, such as, for example, machine language or assembly
5 language.” *Id.* at 23:9-12.

6 54. This disclosure once again firmly links the term “kernel” in the ’768 patent to
7 “program code for interpreting high-level code, commands, and/or instructions supplied by a user
8 or a script into low-level code, such as, for example, machine language or assembly language,” as
9 recited in NVIDIA’s proposed construction.

10 55. Although the citation uses the term “typically,” all of the specification’s examples
11 of software programs for evaluating mathematical expressions that can be used with the alleged
12 invention comprise kernels that interpret high level code into low level code.

13 56. Next, the specification uses the term “kernel space” in the context of operating
14 systems when it describes “an operating system, that divides memory and/or other system
15 resources” “into a user space, a kernel space, an application space (for example, a portion of the
16 user space allocated to an application program), and/or an operating system space (for example, a
17 portion of the user space allocated to an operating system).” ’768, col. 25:48-54.

18 57. In this passage, “kernel space” refers to the usual division of computer memory
19 with dedicated memory space allocated for critical operating system kernel operations, while
20 other memory partitions are allocated to the user, applications, and non-kernel portions of the
21 operating system. It is not referring to the claimed “kernel.”

22 58. Finally, the specification includes a paragraph stating: “For purposes of
23 illustration, some embodiments are described herein in the context of cluster computing with
24 Mathematica software. The present disclosure is not limited to a single software program; the
25 systems and methods can be used with other application software such as, for example, Maple®,
26 MATLAB®, MathCAD®, Apple Shake®, Apple® Compressor, IDL®, other applications
27 employing an interpreter or a kernel, Microsoft Excel®, Adobe After Effects®, Adobe
28 Premiere®, Adobe Photoshop®, Apple Final Cut Pro®, and Apple iMovie®.” ’768, col. 4:14-25.

1 59. A POSITA evaluates this statement in relation to the claimed subject matter. The
2 claims at issue all require a single-node kernel that evaluates mathematical expressions and
3 interpret user instructions. Of the third party application programs recited here, Mathematica,
4 Maple, MATLAB and MathCAD are directed to evaluation of mathematical expressions, and
5 comprise a single-node kernel that can perform such evaluation by interpreting user instructions.
6 A POSITA would not consider the other applications as examples of the claimed kernel because
7 they are not directed to evaluating mathematical expressions, and, therefore, do not comprise an
8 interpreter kernel for mathematical expression evaluation.

9 60. Moreover, although this family of patents includes claims that are limited to
10 Mathematica, the claims asserted here are not so limited. Apart from the high-level concept that
11 the “present disclosure” is not necessarily limited to Mathematica, nothing in this passage
12 redefines “kernel” in a way contrary to the remainder of the written description, points to any
13 specific feature in any program as constituting a “kernel” or not constituting a “kernel,” or
14 addresses any of the features of the claims in this case, including the requirement that the kernel
15 evaluate mathematical expressions.

16 61. The preferred embodiment is directed to Mathematica, but all of the other well-
17 known mathematical software packages recited in the specification operate in a similar manner.
18 In order to solve a mathematical expression, such as an equation, a user writes that expression as
19 a set of high-level instructions (e.g., Mathematica, Maple, MATLAB or MathCAD instructions).
20 The high-level instructions are then interpreted by these programs into low-level code for
21 execution.

22 62. Interpreters are advantageous in mathematical programs because they are easy to
23 learn and allow users to flexibly input and modify their mathematical problems. For this reason,
24 Mathematica, Maple, MATLAB and MathCAD all have interpreter kernels. Interpreting is
25 generally not a one-time process. Instead, an interpreter converts high-level code into low-level
26 code line by line every time the code is input and run. The American Heritage College Dictionary
27 (3rd ed. 1993), for example, defines “interpreter” as “a program that translates an instruction into
28 a machine language and executes it before the next instruction.” This commonly accepted

1 definition is consistent with the description of “interpreting” and “interpreter” in the ’768 patent
2 specification, where, as I have shown, these terms refer to conversion of high-level instructions
3 into low level (machine language) instructions.

4 63. Accordingly, the plain claim language, the uniform and clear teaching of the
5 specification, and the knowledge in the relevant art all teach that the claimed “kernel” is an
6 interpreter that evaluates mathematical expressions by interpreting high-level user instructions
7 into low level code for execution by the processor running the kernel. This understanding is
8 reflected in NVIDIA’s proposed claim construction.

9 **C. “single-node kernel” (claims 1, 26, 29, 35)**

10 64. In the ’768 patent claims, the term “kernel” appears in the phrase “single-node
11 kernel.” In the claims, each node accesses its own single-node kernel: “wherein each of the nodes
12 is configured to access a non-transitory computer-readable medium comprising program code for
13 a single-node kernel.” ’768, col. 30:28-32. In my opinion, for the reasons I discuss below, the
14 term “single-node kernel” is understood by a POSITA to mean: “*a kernel that is designed to*
15 *communicate with and run on only a single node.*”

16 65. As I discuss above, the purpose of the ’768 patent’s alleged invention is to extend
17 the capability of third party mathematical software, such as Mathematica. In explaining the
18 problem it trying to solve, the specification states: “Many computer application programs are not
19 currently designed to benefit from advantages that computer clusters can offer, even though they
20 may be running on a group of nodes that could act as a cluster. Some computer programs can run
21 on only a single node because, for example, they are coded to perform tasks serially or because
22 they are designed to recognize or send instructions to only a single node.” ’768, col. 1:22-29.

23 66. Thus, at the highest level, the stated problem is of adapting software that was
24 designed to “run on only a single node” and “recognize or send instructions to only a single node”
25 so that it can instead run as part of a cluster of nodes to perform cluster computing and recognize
26 and send instructions for multiple nodes.

27 67. The specification further explains that the problem to be solved is extending
28 cluster computing capability to software with kernels that run on, and communicate with, only on

1 a single node when it states: “Some software programs include *a kernel that is designed to*
 2 *communicate with a single node*. An example of a software package that includes a kernel that is
 3 designed to communicate with a single node is Mathematica® from Wolfram Research, Inc.
 4 (“Mathematica”).” ’768, col. 1:37-43. The specification further explains: “In one embodiment,
 5 one or more software modules such as kernels, run on nodes within the interconnected computer
 6 systems. In one embodiment, the kernels are designed to run on only a single node.” *Id.* at 4:56-
 7 59. As I discuss above, the Mathematica kernel interprets user instructions into low-level machine
 8 code for only a single node. Therefore, the kernel runs on a single node, and communicates its
 9 output in the form of low-level code to the same node for execution.

10 68. The single-node kernels are further illustrated in Fig. 2, which shows kernels 206a-
 11 d each running on, and communicating with, only a single node. ’768, col. 5:33-38. The
 12 specification purports to solve this alleged inability of the single-node kernels to run on more than
 13 one node by introducing program code to interconnect the single-node kernels, as illustrated in
 14 Fig. 2.

15 69. In light of the consistent and uniform teaching of the specification and the
 16 knowledge in the relevant art, a POSITA would understand “single-node kernel” to mean “*a*
 17 *kernel that is designed to communicate with and run on only a single node.*”

18 **D. “the first node” (claim 35)**

19 70. Claim 35 recites “[a] computer cluster node,” “a second node,” and “a third node,”
 20 but then it also recites “*the* first node.” ’768, col. 35:9, 35:28, 35:38, 36:4 (emphasis added). In
 21 my opinion, for the reasons I discuss below, the term “the first node” and the limitation within
 22 which it appears (’768, col. 35:37-36:4) are indefinite.

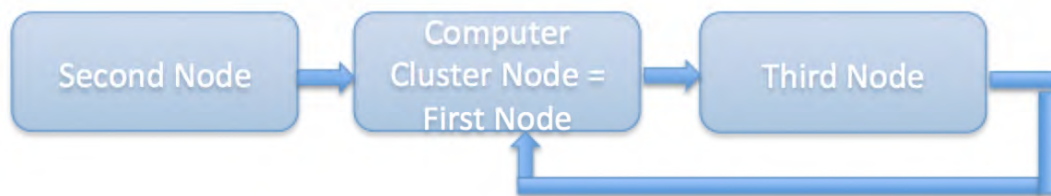
23 71. “[T]he first node” lacks antecedent basis and a POSITA would not know with
 24 reasonable certainty whether the first node refers to the originally recited “[a] computer cluster
 25 node,” any of the other nodes previously recited in the claim, or another fourth node.

26 72. Claim 35 recites the following communication scheme:



73. According to claim 35, the claimed “computer cluster node” “receive[s] calls” from a second node (’768, col. 35:28-36), evaluates a “first mathematical expression” (*id.* at 35:35-36) and communicates the evaluation results to “a third node” (*id.* at 35:37-40). The third node receives the evaluation result and executes a second mathematical expression evaluation using the received result. *Id.* at 35:40-36:3. The third node then communicates a result of its evaluation “to the first node.” *Id.* at 36:3-4. The term “the first node” appears for the first time in the claim at col. 36:3-4 and lacks antecedent basis.

74. Given the lack of antecedent basis for “the first node,” the phrase “communicate a result of the second mathematical expression evaluation to the first node” fails to inform a POSITA with reasonable certainty about the scope of claim 35. A POSITA can assign several equally reasonable interpretations to the claim language. For example, a reasonable reading of the claim language is that “the first node” refers to the first of the nodes recited in the claim, i.e., the “computer cluster node.” ’768, col. 35: 10-11. This sequence of communications is illustrated in the following graphic:



75. Here, the “computer cluster node” is “the first node,” which is why the middle rectangle representing this node is labeled “Computer Cluster Node = First Node.” This computer cluster node / first node “communicate[s] a result of the first mathematical expression evaluation to a third node,” as shown by the arrow from Computer Cluster Node to Third Node. ’768, col. 35:38-39. The third node then “execute[s] at least a second mathematical expression evaluation using the result of the first mathematical expression evaluation.” *Id.* at 35:42-36:3. The third node then communicates the result of the second evaluation back to the computer cluster node / first node *Id.* at 36:3-4. A POSITA would think this communication reasonable at least because the

1 third node is evaluating a mathematical expression using the results received from the computer
 2 cluster node. Sending the result back to the “computer cluster node” could allow that node to
 3 perform further evaluation using the results received from the third node.

4 76. Alternatively, a POSITA can equally conclude that “the first node” is yet another
 5 fourth node (in addition to “the computer cluster node,” “the second node” and “the third node”)
 6 that receives results from the computer cluster node and performs yet more mathematical
 7 evaluations. The following diagram illustrates this interpretation:



10
 11 77. Such a scenario is consistent with the claim language. It is also possible in the case
 12 of complex mathematical evaluations that may require multiple rounds of intermediate results and
 13 a large number of nodes. Nothing in claim 35 precludes such a system with four or more nodes in
 14 communication with each other.

15 78. Fig. 2, for example, illustrates a cluster of five nodes, with “cluster node modules”
 16 204 providing communications between the nodes. The specification teaches that
 17 “[c]ommunications can occur between any two or more cluster node modules.” ’768, col. 6:9-10.
 18 The specification also asserts that “[i]ntercommunication among kernel modules 206 a-e during
 19 thread execution, which is made possible by cluster node modules 204 a-e, provides advantages
 20 for addressing various types of mathematic and scientific problems, for example.” *Id.* at 6:26-29.
 21 Moreover, the written description does not disclose any specific order of operations.

22 79. I understand that Plaintiff ACS has proposed that “the first node” is “the second
 23 node,” as illustrated by the following diagram:



80. I disagree that a POSITA would reach this conclusion, or any other conclusion, with reasonable degree of certainty regarding the scope of claim 35.

81. In fact, the suggestion to interpret the claim phrase “the first node” as meaning “the second node” merely highlights the lack of reasonable certainty regarding claim scope because it suggests yet another possible interpretation of this claim term. As I have explained, a POSITA can instead reasonably conclude that “the first node” is the first node recited in the claim or some other fourth node that needs the result from the claimed node to perform further evaluations of mathematical expressions.

82. Although claim 35 recites that the second node is “configured to access program code for a user interface” (‘768, col. 35:28-32), this feature does not resolve the ambiguity. The limitation at issue does not recite that “the first node” is any way associated with the previously recited code for a user interface. To the contrary, the limitation at issue says only: “communicate a result of the second mathematical expression evaluation to the first node.”

83. Nor does the following limitation resolve the ambiguity. That limitation recites: “wherein the user connection interface is configured to return at least one result of mathematical expression evaluation to a user interface or script.” By referring to “*at least one* result,” rather than “*the* result” of the first or second mathematical expression evaluation, this limitation does not tie its recited result to any of the preceding results recited in the claim. Moreover, the limitation also does not tie itself to the previously recited “program code for a user interface” associated with the second node, but instead recites “a user interface or a script” generally. Finally, the limitation refers to the “at least one result” as coming from “the user connection interface.” The user-connection-interface element is recited as being included with the original “computer cluster node,” not the third node (which is the node that sends the result recited in the

1 limitation at issue).

2 84. I have reviewed the prosecution history of the '768 patent. As-issued claim 35
3 (claim 37 during prosecution) was the result of an examiner amendment, which was approved by
4 the applicants. 11/29/2018 Notice of Allowability and Examiner Amendment at 12; 11/21/2018
5 Examiner-Initiated Interview Summary. This amendment introduced the “second,” “third” and
6 “first” nodes to the claim language. While the examiner stated that the amendment’s intent was to
7 “make independent claim 37 have all the limitations from the other independent claims,” this does
8 not sufficiently inform a POSITA regarding what “the first node” may be in the claim in view of
9 the multiple equally reasonable interpretations I discuss above and the many differences between
10 the express language of claim 35 and the other independent claims.

11 85. Adding to the confusion is the fact that the previous version of claim 37 also
12 recited “a first node:”

13 program code that, when executed, is capable of causing the hardware processor
14 to receive calls from a first node, execute at least a first mathematical expression
15 evaluation, and communicate a result of mathematical expression evaluation to another
16 node for use in at least a second mathematical expression evaluation;

17 02/14/2014 Amendment at 10.

18 86. The claimed node received calls from this “first node” but sent results to “another
19 node,” indicating that this “first node” is not the first node of the other claims, which received
20 results. Was the “first node” in the later 11/29/2018 Examiner Amendment intended to be like
21 this previous “first node” or something different? Again, a POSITA would not be able to reach
22 any reasonably certain conclusion about claim scope, further confirming my opinion that the
23 claim language is indefinite.

24 **E. “peer-to-peer architecture” (claims 1, 35)**

25 87. The term “peer-to-peer architecture” appears in claims 1 and 35. In my opinion,
26 the term “peer-to-peer architecture” means “*an architecture in which each node is configured to*
27 *communicate with other nodes.*”

28 88. A POSITA would understand that a “peer-to-peer architecture” is one that supports

1 communications between the nodes of the cluster, in contrast to the gridMathematica and PCT
2 prior art where the nodes allegedly could not intercommunicate. (“In contrast, a parallel
3 computing toolkit (PCT) that is provided as part of the gridMathematica software package does
4 not provide a means for instances of the same code running on different nodes to communicate,
5 collaborate, or coordinate work among the instances.” ’768, col. 12:26-30).

6 89. In this regard, the ’768 patent specification explains that a number of MPI protocol
7 mechanisms “provide[] the peer-to-peer behavior” and that “[v]ia this mechanism, code running
8 within multiple, simultaneously running kernel modules (for example, Mathematica kernels) can
9 interact on a pair-wise or collective basis, performing calculations, processing, or other work on a
10 scale larger and/or faster than one kernel could have done alone.” ’768, col. 25:37-44. A POSITA
11 understands that this MPI mechanism refers to the claimed “mechanism for the nodes to
12 communicate results of mathematical expression evaluation with each other using a peer-to-peer
13 architecture.” ’768, col. 30:33-35. Accordingly, the mechanism implements the communication
14 scheme in the peer-to-peer architecture where cluster nodes “can interact on a pair-wise or
15 collective basis.” This further informs a POSITA that in the peer-to-peer architecture, each of the
16 nodes can communicate with the other cluster nodes without any restriction on what path the
17 communication may take.

18 90. This understanding of “peer-to-peer architecture” is also consistent with extrinsic
19 evidence. For example, the IEEE dictionary’s definition of “peer-to-peer communication” is
20 “communication between two or more processes or programs by which both computers can
21 exchange data freely.” IEEE 100, The Authoritative Dictionary of IEEE Standards Terms, 7th Ed.,
22 2000 at p. 804. This definition reflects the understanding of a POSITA, and does not require any
23 particular communication path, or exclude any means of direct or indirect communication
24 between nodes. Moreover, a peer-to-peer architecture may have any topology where computers
25 exchange data freely, regardless of whether one or more nodes also serve as a management or
26 root node for the cluster. An architecture where each of the nodes can communicate with the
27 others, as illustrated in ’768 Fig. 2, is one where “two or more processes or programs by which
28 both computers can exchange data freely.”

1 91. Accordingly, in my opinion, a POSITA would understand the term “peer-to-peer
2 architecture” to mean “an architecture in which each node is configured to communicate with
3 other nodes.”

4 92. I understand ACS has proposed the following different construction of this term:
5 “an architecture in which each node can communicate tasks and data with other nodes without the
6 tasks and data being required to go through a central server or master node.” I disagree that a
7 POSITA would have understood this term to have that meaning.

8 93. In my opinion, the intrinsic evidence does not support the construction proposed
9 by ACS.

10 **Claim Language**

11 94. Starting with the claim language, the phrase “peer-to-peer architecture” first
12 appears in claims 1 and 35 in a limitation relating to a communication mechanism that permits the
13 nodes to communicate with each other. For example, the full limitation in claim 1 recites: “a
14 mechanism for the nodes to communicate results of mathematical expression evaluation with
15 each other using a peer-to-peer architecture.” ’768, col. 30:33-35. This phrase informs a POSITA
16 what is being communicated using the peer-to-peer architecture – “results of mathematical
17 expression evaluation.” (As I discuss below, in my opinion the terms “a mechanism for the nodes
18 to communicate...” in the asserted claims are written in a means-plus-function format.) Another
19 type of communication that the claims associate with the peer-to-peer architecture are user
20 instructions communicated by one or more nodes. ’768, col. 30:65-67.

21 95. ACS’s proposed construction requires communication of “tasks and data,” but
22 neither of these terms is found in the claims. A POSITA would not understand these two things to
23 be the same as the two things the claims do recite – user instructions and results of mathematical
24 expression evaluation. A POSITA would therefore not understand this term to carry the meaning
25 proposed by ACS rather than what the claims actually recite.

26 96. Likewise, ACS’s construction recites “a central server or master node,” but these
27 terms also are not recited in the claims. (In fact, the term “central server” is not recited anywhere
28 in the specification, either.) Claim 1, for example, recites first, second and third nodes. Again, a

1 POSITA would not understand this term to substitute these extraneous words into the claims over
2 what actually is recited on the claims.

3 97. Another aspect of ACS's proposed construction is that the communication can be
4 "without the tasks and data being required to go through a central server or master node." A
5 POSITA would not find this limitation supported by the claim language because nothing in the
6 term "peer-to-peer" as used in cluster computing requires any particular communication path
7 between nodes, so long as one peer can communicate with another peer. The claim language
8 likewise does not impose a restriction on how any of the communications associated with the
9 peer-to-peer architecture may occur.

10 **Written Description**

11 98. A POSITA also would not understand the specification to support ACS's proposal.

12 **In Preferred Embodiments, Tasks Go Through Master Node**

13 99. ACS's proposal would exclude any embodiment where tasks or data are required
14 to go through "a central server or master node." However, preferred embodiments of the '768
15 patent comprise just such a "master node," referred to as the "root" or "root processor" in the
16 specification.

17 100. The cluster in Fig. 2 has one node with which the user interacts through a user
18 interface – the node comprising kernel module 206a and user interface module 202. This node is
19 often referred to as a "master" or "root" node in the cluster. The '768 patent specification
20 repeatedly discloses a "root processor" sending expressions for evaluation by other nodes. For
21 example, the mpiBcast instruction communicates "[a]n expression [that] is expected to be
22 supplied by the root processor." '768, col. 14:45-65. This root processor is a root or master node
23 supplying an expression to the other nodes with the mpiBcast call. (In claims 1, 26 and 29, the
24 "first node" comprises a user interface and distributes calls, functions that a POSITA associates
25 with a master or root node.)

26 101. In the preferred embodiments, tasks go through this master / root node. The
27 specification uses the term "tasks" in the following sentence: "MPI calls and advanced cluster
28 commands are used to parallelize program code received from an optional user interface module

1 208 and distribute tasks among the kernel modules 206a-e.” ’768, col. 6:15-19. This sentence
 2 describes the behavior of the node comprising cluster node module 204a and kernel module 206a,
 3 because this node has the user interface, as illustrated in Fig. 2.¹ ’768, col. 6:9-25. Therefore, a
 4 POSITA understands that tasks are distributed by this “root” or “master” node (so-called because
 5 it has the user interface, as I explain above) to other nodes.

6 102. The next sentence of the specification states that “[t]he cluster node modules 204a-
 7 e provide communications among kernel modules 206a-e while the tasks are executing.” ’768,
 8 col. 6:19-21. This sentence tells a POSITA that the nodes may inter-communicate while
 9 executing the tasks sent to them by the root or master node. However, there is no teaching that
 10 tasks (rather than intermediate results of mathematical expression evaluation, for example) are
 11 inter-communicated by the nodes. In fact, the only teaching is that tasks are communicated by the
 12 node with the user interface 202 – the master or root node. This is directly contrary to ACS’s
 13 proposed construction, which requires tasks to be communicated without being required to go
 14 through the master node.

15 103. The specification further confirms this understanding regarding tasks in a section
 16 titled “Cluster Node Module Operation.” ’768, col. 24:35-25:21; *id.* at Fig. 5. This section
 17 describes the operation of cluster node module 204a in Fig. 2, which, as I explain above, is
 18 located on the root/master node comprising user interface 202. “[W]hen a user enters a command
 19 in the user interface module 202, the cluster node module 204a connected to the user interface
 20 module 202 submits the user command to all other cluster node modules 204b-e in the computer
 21 cluster.” ’768, col. 24:54-58. These commands include, for example, “calls to MPI from within
 22 the user interface module 202.” *Id.* at 24:61-62. Once again, this disclosure teaches that
 23 mathematical evaluation tasks based on commands input by the user are communicated by the
 24 root / master node as calls to the other nodes. ’768, col.17:1-45 (The specification uses the term
 25 “function call” or “call” in reference to commands implementing a task, e.g., the call
 26 ParallelTable is a command implementing a table parallelization task on a cluster.). There is no

27 ¹ This sentence misidentifies user interface module 202 as 208. Module 208 in Fig. 2 is the cluster
 28 configuration module.

disclosure of tasks being communicated without going through this node, contrary to ACS's construction.

Load Balancing Requires Tasks To Go Through Master Node

104. Moreover, ACS's construction is contrary to a preferred embodiment disclosed in the specification that uses load balancing. As described in the specification, to accomplish load balancing, the "root processor" – i.e., the master node – is responsible for assigning mathematical expression evaluation tasks to each of the cluster nodes. Specifically, the "root processor assigns a small subset of the possible calls for a function to each processor on the cluster." '768, col. 21:23-26. "[T]he root processor will continue assigning additional work to processors as they become available." *Id.* at 21:25-30. As I explain above, calls to evaluate a mathematical function are referred to as tasks assigned to the cluster nodes.

105. As an example, the root processor assigns tasks to evaluate functions f[1] to f[100] to a group of four processors "until all results are calculated." *Id.* at 21:34-42. The specification teaches that "[o]ne could implement this by assigning f[1], f[2], f[3], f[4] to each of processors 0 (the root can assign to oneself) through 3. If the f[2] result came back first, then processor would be assigned f[5]... The assignments continue until all results are calculated." *Id.* This description teaches that the root processor (processor 0) assigns all of the tasks to evaluate mathematical expressions to the other cluster nodes. The specification also teaches assigning subsets of calls by delegation, where "one processor node may not necessarily be in direct control of the other processors." '768, col. 21:45-50. However, even in this case, a POSITA understands that the initial assignment of subsets in the "hierarchy of assignments" is done by the root / master node, which sits at the top of the hierarchy and controls the workload of the overall cluster. *Id.*

106. Because the root / master node assigns all of the tasks, each of the tasks is required to "go through" the master node, contrary to ACS's proposal.

107. In fact, communication of tasks by anything other than the root or master node does not make sense in view of this load balancing scheme; for this load balancing scheme to function correctly, the root / master node is the exclusive distributor of tasks in the cluster.

Results Are Communicated Between Nodes

108. The '768 specification differentiates between tasks and mathematical evaluation results. For example, in the column 6 disclosure discussed above, the specification teaches that "tasks" are distributed" by the node with the user interface module (i.e., the root or master node), while the "results of evaluations" are communicated back to that node by the other nodes. '768, col. 6:15-25.

109. While results are not tasks, they are a type of data communicated between nodes. The claims recite, for example, a second node "configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of mathematical expression evaluation to a third node." '768, col. 30:44-50. Similarly, while the specification teaches tasks communicated from a root or master node to other nodes, it teaches that data such as results are communicated between any of the nodes: "[O]nce the kernel module 206 completes its evaluation, it returns the kernel module's output to the cluster node module 204 to which it is connected. Depending on the nature of the result from the kernel module, the cluster node module 204 can report the result to a local computer system or pass the result as a message to another cluster node module 204." '768, col. 25:9-21; *see also id.*, col. 27:36-37 (data communicated between nodes).

Specification Distinguishes Prior Art Based on Data Communications, Not Tasks

110. It is this ability to communicate result data between nodes that the specification asserts distinguishes the alleged invention from the prior art gridMathematica product. The '768 patent specification criticizes this gridMathematica prior art because its nodes "generally do not communicate with one another as peers." '768, col. 1:44-47. Instead: "One kernel is designated the master kernel, which handles all input, output, and scheduling of the other kernels (the computational kernels or slave kernels). Computational kernels receive commands and data only from the node running the master kernel. Each computational kernel performs its work independently of the other computational kernels and *intermediate results* of one job do not affect other jobs in progress on other nodes." '768, col. 1:5-62 (emphasis added).

111. This is contrasted with the alleged invention, where “[i]ntercommunication among kernel modules 206a-e during thread execution, which is made possible by cluster node modules 204a-e, provides advantages for addressing various types of mathematic and scientific problems, for example.” *Id.* at 6:26-29.

112. In the same vein, the ’768 patent specification criticizes the prior art PCT parallel computing kit for Mathematica on the basis that it “provides commands that connect Mathematica kernels in a master-slave relationship rather than a peer-to-peer relationship as enabled by some embodiments disclosed herein.” ’768, col. 12:30-33. While this passage purports to differentiate “some embodiments” as having a “peer-to-peer relationship” rather than a “master-slave relationship,” there is no teaching that the invention excludes a “master-slave relationship” when communicating tasks. Moreover, as I discuss herein, the preferred embodiments also have a root or master node, and it is this root / master node that is responsible for distributing calls to the other nodes. Accordingly, a POSITA would not understand this disclosure to exclude clusters with a master node, or clusters where the master node communicates tasks. At most, the specification distinguishes clusters where peer nodes cannot pass result messages to each other in any manner, whether directly or through the root/master node.

113. As I discuss above, a POSITA would not understand the alleged invention to differ from this prior art based on how tasks are distributed – the root or master node in the preferred embodiments communicates tasks to the other nodes, similarly to how the specification describes the prior art. On the other hand, a POSITA would conclude based on the specification that the ability to communicate intermediate results between cluster nodes while the nodes are computing is the alleged improvement over the prior art.

There is No Limitation on How Data is Sent

114. Even with regard to the communication of intermediate results, there is no indication in the claims or the specification requiring the routing of intermediate results sent from one node to another through a root/master node is excluded. As quoted above, peer-to-peer communications do not exclude any methods of implementing the inter-communication, as shown, for example, in the IEEE dictionary’s definition of “peer-to-peer communication” as

“communication between two or more processes or programs by which both computers can exchange data freely.” IEEE 100, The Authoritative Dictionary of IEEE Standards Terms (7th ed. 2000), at 804. Peer cluster nodes can exchange data freely through direct messaging, but they can also do so by routing messages through the cluster’s root or master node.

115. Nor does a “peer-to-peer architecture” exclude the notion of a root or master node. This is reflected in the IEEE dictionary definition of “peer-to-peer communication” –peer nodes can exchange data freely even if one of the nodes also acts as a root / master node. Accordingly, a peer-to-peer architecture that supports peer-to-peer communication can exist unless a cluster is set up without the ability for the nodes to intercommunicate, regardless of whether there is a master node (the ’768 patent’s description of the prior art gridMathematica and PCT).

116. In fact, as I have explained, the ’768 patent preferred embodiment comprises just such a root / master node. For example, the following table lists a number of functions for performing parallel computation, most of which include an argument **root**, signifying a “processor whose ID is root”:

TABLE 1

Call	Description
ParallelDo[expr, loopspec]	Like Do[] except that it evaluates expr across the cluster, rather than on just one processor. The rules for how expr is evaluated is specified in loopspec, like in Do[].
ParallelFunctionToList[f, count]	Evaluates the function f[] from 1 to count, but across the cluster, and returns these results in a list. The third argument has it gather this list into the processor whose ID is root.
ParallelFunctionToList[f, count, root]	
ParallelTable[expr, loopspec]	Like Table[] except that it evaluates expr across the cluster, rather than on just one processor, returning the locally evaluated portion. The third argument has it gather this table in to the processor whose ID is root.
ParallelTable[expr, loopspec, root]	
ParallelFunction[f, inputs, root]	Like f[inputs] except that it evaluates f on a subset of inputs scattered across the cluster from processor root and gathered back to root.
ParallelNIntegrate[expr, loopspec]	Like NIntegrate[] except that it evaluates a numerical integration

1 '768, col. 17:20-45.

2 117. For example, the function ParallelFunction[f, inputs, root] evaluates a function f
3 “on a subset of inputs scattered across the cluster from processor root and gathered back to root.”
4 *Id.* This function tells the cluster nodes to evaluate inputs that the nodes receive from the root /
5 master node and then send the results back to the root node. These disclosures inform a POSITA
6 that the claimed cluster with a peer-to-peer architecture also comprises a root or master node.
7 Moreover, as I discuss above, embodiments of the alleged invention implement load balancing,
8 wherein the root / master node distributes calls to other nodes, e.g., where “the root processor
9 assigns a small subset of the possible calls for a function to each processor on the cluster.” ’768,
10 col. 21:23-25.

11 118. For all of these reasons, the claims, the specification and the knowledge of a
12 POSITA do not support the idea that a “peer-to-peer architecture” entails any restriction on how
13 nodes communicate, and there is no teaching that they do so “without the tasks and data being
14 required to go through a central server or master node.”

15 **F. “*mechanism for the nodes to communicate results of mathematical expression*
16 *with each other*” terms (claims 1, 26, 29, 35)**

17 119. Each of the asserted claims 1, 26, 29, and 35 includes a variant of the limitation “a
18 mechanism for the nodes to communicate results of mathematical expression with each other.” In
19 connection with my analysis of these terms, I have been informed by counsel for NVIDIA that a
20 limitation may be written in a “means-plus-function” format under 35 U.S.C. § 112, ¶ 6, even
21 when it does not include the words “a means for.” When a claim term lacks the word “means,”
22 the rebuttable presumption that the term is not in the means-plus-function format can be
23 overcome and § 112, ¶ 6 will apply if a party demonstrates that the claim term fails to recite
24 sufficiently definite structure or else recites function without reciting sufficient structure for
25 performing that function. In undertaking this analysis, we ask if the claim language, read in light
26 of the specification, recites sufficiently definite structure to avoid § 112, ¶ 6. Generic terms such
27 as “mechanism,” “element,” “device,” and other nonce words that reflect nothing more than
28 verbal constructs may be used in a claim in a manner that is tantamount to using the word

“means” because they typically do not connote sufficiently definite structure. If § 112, ¶ 6 applies, the claims are construed to cover only the structure, materials, or acts described in the specification as corresponding to the claimed function and equivalents thereof.

120. My opinion, based on my analysis under the framework outlined above, is that a POSITA understands the “mechanism for...” terms to follow the “means-plus-function” format. The function and corresponding structure for these terms are as follows:

Term	Function	Structure
“a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture” (claim 1)	Communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture	Message-Passing Interface (“MPI”) module 302, RMQ received message queue 306 and MRQ message receiving queue 308 configured to execute the process described at 25:29-43
“a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using asynchronous calls” (claim 26)	Communicate results of mathematical expression evaluation with each other using asynchronous calls	Message-Passing Interface (“MPI”) module 302, including mpiSend, mpiRecv, and mpiTest commands as described at 14:1-28, RMQ received message queue 306 and MRQ message receiving queue 308 configured to execute the process described at 25:29-43
“a mechanism for the nodes to communicate results of mathematical expression evaluation with each other” (claim 29)	Communicate results of mathematical expression evaluation with each other	Message-Passing Interface (“MPI”) module 302, RMQ received message queue 306 and MRQ message receiving queue 308 configured to execute the process described at 25:29-43
“a mechanism to communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture” (claim 35)	Communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture	Message-Passing Interface (“MPI”) module 302, RMQ received message queue 306 and MRQ message receiving queue 308 configured to execute the process described at 25:29-43

121. While the four “mechanism for...” limitations have some variations, there is a lot of commonality in the claim language, including the surrounding limitations. For conciseness, I describe my opinions with regard to claim 1, and note claim-to-claim variations as appropriate.

122. In my opinion, the term “a mechanism for the nodes to communicate results of

1 mathematical expression evaluation with each other using a peer-to-peer architecture” fails to
2 recite a sufficiently definite structure from the perspective of a POSITA. The term “a mechanism”
3 in itself does not have a structural meaning in the relevant computer art. Considering the longer
4 phrase “a mechanism for the nodes to communicate results of mathematical expression evaluation
5 with each other,” it also does not connote a structure to a POSITA, and does not have a generally
6 accepted meaning. Instead, it merely recites a “black box” for performing the function of
7 “communicat[ing] results of mathematical expression evaluation” between the nodes. Stated
8 differently, a POSITA does not understand the recited “mechanism for...” to be a stand-in for a
9 definite structure, such as an electrical circuit, or any other definite structure. Rather, the terms
10 merely recite a communication function.

11 123. The addition, in claims 1 and 35, of the requirement that the communicating be
12 done using a “peer-to-peer architecture” does not impart sufficient structure where the rest of the
13 limitation provides none. As I discuss above, the correct construction of “peer-to-peer
14 architecture” is “an architecture in which each node is configured to communicate with other
15 nodes.” This architecture in itself is not a sufficiently definite structure. Instead, as discussed
16 above, it is a generalized methodology of intercommunication between cluster nodes, rather than
17 a structure for implementing that methodology.

18 124. The same conclusion holds under ACS’s proposed construction of “peer-to-peer
19 architecture.” With that construction, the full limitation reads “a mechanism for the nodes to
20 communicate results of mathematical expression evaluation with each other using an architecture
21 in which each node can communicate tasks and data with other nodes without the tasks and data
22 being required to go through a central server or master node.” The construction doesn’t add any
23 definiteness to the missing structure of the recited “mechanism.” Instead, it merely restricts an
24 undefined architecture used by an undefined mechanism from requiring that tasks and data “go
25 through a central server or master node.” The overall effect is still of functional language, and not
26 a sufficiently definite structure for performing that function.

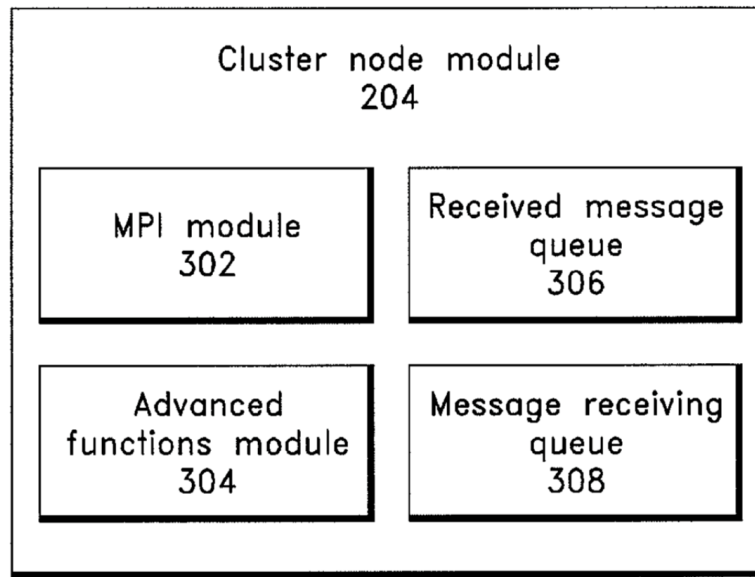
27 125. The same analysis and conclusion apply to the “mechanism for...” limitations in
28 claims 26 and 29, which do not recite the “peer-to-peer architecture” limitation. Claim 26 recites

1 “using asynchronous calls” for the communication, but this language is also purely functional –
2 no structure for communicating results of mathematical expression evaluation using
3 asynchronous calls is recited. As discussed below, the “asynchronous calls” disclosed in the
4 specification are certain MPI *functions*, which can be referred to as “calls.” A POSITA would not
5 understand the addition of a generic recitation of asynchronous calls – functions – to add
6 sufficiently definite structure to the functional language of the rest of the limitation.

7 126. For these reasons, it is my opinion that a POSITA would consider the “mechanism
8 for...” claim terms to recite a function, and, therefore, be written in a “means-plus-function”
9 format under 35 U.S.C. § 112, ¶ 6.

10 127. A POSITA would consider the claimed function to comprise the language of the
11 limitation. For example, in claim 1, the function is “communicate results of mathematical
12 expression evaluation with each other using a peer-to-peer architecture.”

13 128. With regard to the corresponding structure, as I discuss above, the alleged
14 advantage of the ’768 invention over the prior art is the ability of the kernels 206 in Fig. 2 to
15 intercommunicate and send mathematical expression evaluation results to each other. This
16 “[i]ntercommunication among kernel modules 206a-e during thread execution, which is made
17 possible by cluster node modules 204a-e, provides advantages for addressing various types of
18 mathematic and scientific problems.” ’768, col. 6:26-29. Structures within the cluster node
19 module perform the function of communicating results of mathematical expression evaluation. *Id.*
20 at 6:3-8, 6:19-21, 6:26-29, 25:9-16. The cluster node module includes the four modules shown in
21 Fig. 3:



129. Of these four modules, the MPI module 302, received message queue 306 and message receiving queue 308 are disclosed as the structure of the mechanism for communicating mathematical evaluation results using a peer-to-peer architecture. In the cluster node module, the structure performing the node-to-node communication functions, including the communication of mathematical evaluation results, is the software module implementing MPI (Message Passing Protocol). '768, col. 12:4146. No other mechanism for communicating results between nodes is disclosed. "The first cluster node module 204a is in communication with one or more other cluster node modules 204b, 204c, and so forth, each of which provides a set of MPI calls and/or advanced cluster commands. In one embodiment, MPI may be used to send messages between nodes in a computer cluster." '768, col. 6:3-8. "

130. The MPI Module 302 implements MPI calls "that send data, equations, formulas, and/or other expressions. Simply sending expressions from one node to another is possible with these most basic MPI calls. One node can call to send an expression while the other calls a corresponding routine to receive the sent expression." '768, col. 13:15-21. The specification recites specific MPI calls in the MPI Module 302 for implementing communications between nodes, including mpiSend, mpiRecv; their non-blocking or asynchronous counterparts mpiIsend, mpiIRecv; and mpiTest, used in the receiving process. '768, col. 13:25-14:28. MPI Module 302

1 also includes collective MPI calls, which “can also provide commonly used *mechanisms to send*
 2 *expressions* between groups of nodes.” ’768, col. 14:35-15:9 (emphasis added). Thus, the first of
 3 only two instances where the specification expressly addresses the “mechanism” for
 4 communication, it is in reference to MPI Module 302. These disclosures inform a POSITA that
 5 MPI Module 302 forms a part of the structure for the claimed “mechanism to communicate
 6 results...”

7 131. The second express description of the communication mechanism structure relates
 8 to the operation of the RMQ (received message queue) and MRQ (message receiving queue),
 9 which, together with MPI Module 302, implement the MPI communication mechanism. ’768, col.
 10 25:29-43. To receive and process a communication message from other nodes, the
 11 communications are “forwarded to a received message queue (“RMQ”).” *Id.* at 25:29-31. Data
 12 from each entry in the MRQ “is matched with entries in the RMQ,” which allows the node to
 13 determine that the communication is successfully completed. *Id.* at 25:31-36.

14 132. In the next sentence, the specification clearly links the MRQ and RMQ structures
 15 and the algorithm they implement to the claimed function for communicating:

16 *This process provides the peer-to-peer behavior of the cluster node*
 17 *modules 204 a-e. Via **this mechanism**, code running within*
 18 *multiple, simultaneously running kernel modules (for example,*
 19 *Mathematica kernels) can interact on a pair-wise or collective basis,*
 20 *performing calculations, processing, or other work on a scale larger*
and/or faster than one kernel could have done alone. In this manner,
user-entered instructions and data specifying what work will be
done via user commands can be executed more quickly and/or
reliably.

21 ’768, col. 25:37-43 (emphasis added).

22 133. A POSITA understands that the disclosed structures for performing node-to-node
 23 communications, the MPI Module 302, and the MRQ and RMQ queues executing the process
 24 disclosed at col. 25:29-43 (discussed above), perform the function of communicating all MPI
 25 messages “that send data, equations, formulas, and/or other expressions. ’768, col. 13:15-21. The
 26 “data, equations, formulas, and/or other expressions” include results of evaluation of
 27 mathematical expressions, which are in the form of data, equations or formulas that can be further
 28 evaluated by the cluster nodes or, ultimately, presented to the user. As discussed above, the MPI

1 protocol messages sent and received by the MPI Module 302 and processed by the MRQ and
2 RMQ queues allow communications in the claimed peer-to-peer architecture. Accordingly, the
3 structure for performing the claimed function in claim 1 is the Message-Passing Interface (“MPI”)
4 module 302, RMQ received message queue 306 and MRQ message receiving queue 308
5 configured to execute the process described at 25:29-43.

6 134. Claim 29 recites substantially the same communication function as claim 1, but
7 adds the requirement of communicating using asynchronous calls, and removes the “peer-to-peer
8 architecture” limitation. The specification identifies a number of “asynchronous calls [that] make
9 it possible for the kernel to do work while communications are proceeding simultaneously.” ’768,
10 col. 13:41-46. The primary such calls are *mpiIsend* and *mpiIRecv*, which, together with the
11 *mpiTest* command implement asynchronous send and receive communications. *Id.* at 14:1-28.
12 Accordingly, a POSITA would clearly link the structure implementing these calls in the MPI
13 Module 302 with the structure corresponding to the “communicate... using asynchronous calls”
14 function recited in claim 29.

15 135. Claim 26 does not include the function of communicating “using a peer-to-peer
16 architecture” or the requirement of communicating using asynchronous calls. The recited function
17 is simply “communicate results of mathematical expression evaluation with each other.”
18 However, as explained above, the structure clearly linked to the communication function is MPI
19 Module 302 and the MRQ / MRQ queues and related algorithms. Regardless of whether the
20 function requires communicating peer-to-peer, these structures implement *all* of the
21 communication functionality. Accordingly, the structure for claim 26 is the same as the other
22 claims.

23 136. The analysis for claim 35 is substantially the same as claim 1, as summarized in
24 the table above.

25 137. I understand that ACS has proposed the structure of “messaging modules that
26 support communication using a peer-to-peer architecture” for claims 1, 29 and 35, and
27 “messaging modules that support asynchronous communication using a peer-to-peer
28 architecture.” I disagree that a POSITA would consider this to be the correct structure. The term

1 “messaging modules” does not appear in the specification. The term, as used in ACS’ proposed
 2 construction, does not have an accepted meaning in the art. It’s meaning in ACS’ proposal is
 3 unclear, and divorced from the specific structures, discussed above, that the specification clearly
 4 links to the communication function.

5 **G. “cluster node module” (Claims 1, 26, 29, 35)**

6 138. The term “cluster node module” appears in dependent claim 4 and certain other
 7 dependent claims. In my opinion, “cluster node module” means “*a module running on each*
 8 *cluster node, configured to communicate messages with the single-node kernel on the same*
 9 *node, and with other cluster node modules.*”

10 139. As I discuss above, the specification teaches that single-node kernels, such as the
 11 Mathematica kernel, are designed to run on and communicate with only a single node. ’768, col.
 12 1:25-43. Thus, according to the specification, because such single-node kernels cannot
 13 communicate with one another, they cannot operate in a computer cluster. *Id.*

14 140. The alleged invention seeks to address this problem by interposing a software
 15 module between single-node kernels to implement the missing communication capabilities. As I
 16 discuss in the ’768 Patent Summary section above, to provide the node intercommunication
 17 allegedly missing from the grid-computing prior art, the ’768 patent associates a single-node
 18 kernel module on each node with a “cluster node module.” ’768, col. 6:26-30. As shown in ’768
 19 Fig. 2, each cluster node module 204 communicates with its associated kernel and with other
 20 cluster node modules. ’768, col. 2:24-29, 5:33-55. The cluster node modules implement
 21 communications between the cluster nodes using the MPI (Message-Passing Interface) protocol.
 22 ’768, col. 5:61-64, 6:3-20.

23 141. To enable message passing between single-node kernels in Fig. 2, “each of the
 24 kernel modules 206a-e is in communication with a single cluster node module 204a-e,
 25 respectively.” ’768, col. 5:33-35. A single-node kernel does not communicate with other kernels,
 26 but expects to interact with a user interface from which it receives user instructions and to which
 27 it sends back results. The cluster node module takes advantage of this fact by “masquerading” as
 28 a user interface: “[T]he cluster node modules 204a-e are configured to look and behave like a user

1 interface module 202 from the perspective of a kernel module 206a.” ’768, col. 5:67-6:2. This
2 allows the cluster node module to communicate with its single-node kernel on the one hand, and,
3 on the other hand, to communicate with the other cluster nodes via their respective cluster node
4 modules: “As shown in Fig.2, each of the cluster node modules 204a-e is in communication with
5 each of the other cluster node modules 204a-e... Each of the cluster node modules 204a-e is in
6 communication with respective kernel modules 206a-e.” ’768, col. 5:42-44, 6:12-15.

7 142. In this manner, “[t]he cluster node module creates an illusion that a kernel module
8 is communicating directly with the other kernel modules.” ’768, col. 24:29-31. This illusion is
9 achieved by the cluster node module sending and receiving messages to and from other nodes,
10 and exchanging messages with its single-node kernel: “In one embodiment, each cluster node
11 module (for example, the cluster node module 204a) checks for and responds to messages from
12 other cluster node modules 204b-e and from the kernel module 206a repeatedly until those are
13 exhausted.” ’768, col. 25:23-26.

14 143. In accordance with the specification, a POSITA recognizes that the cluster node
15 module implements message passing between its single-node kernel and kernels on other cluster
16 nodes. To do so, it communicates messages with its kernel, and communicates with the other
17 nodes by exchanging messages with their cluster node modules. Accordingly, “cluster node
18 module” means “a module running on each cluster node, configured to communicate messages
19 with the single-node kernel on the same node, and with other cluster node modules.”

20 144. I understand that ACS has proposed to construe cluster node module as “a module
21 that establishes intercommunication among nodes in a computer cluster and allows exchanging
22 messages among nodes using a peer-to-peer architecture.” I disagree for several reasons. First,
23 this proposal recites “a peer-to-peer architecture.” The term “cluster node module” by itself does
24 not necessarily require a peer-to-peer architecture. Moreover, as explained above, I disagree with
25 ACS’s proposed construction for “peer-to-peer architecture.”

26 145. Second, there is no limitation in the specification on what type of communication
27 architecture the cluster node module enables or what path the communication take. Its role is to
28 allow communication between single-node kernels that are not otherwise able to communicate in

1 a cluster. As stated in NVIDIA's proposed construction, the cluster node module fulfills this role
2 by passing messages between the single-node kernels. My third criticism of ACS's proposal is
3 that this role is not properly reflected in ACS's proposal. The "cluster node module" does not just
4 "establish intercommunication," it actually carries out the intercommunications. In fact, the
5 specification teaches that establishing intercommunication is done by a different "cluster
6 configuration module, which "can establish communication with the cluster node modules 204a-
7 e," and "initialize[] the cluster node modules 204a-e to provide cluster computing support for the
8 computer cluster 100." '768, col. 11:32-40.

VII. RESERVATION OF RIGHTS

146. I reserve the right to supplement the opinions I provide in this declaration, to the extent ACS proposes any new construction(s) or to rebut any new evidence ACS may identify. I further reserve the right to supplement my opinions should ACS modify its proposed constructions for any other claim term at issue.

I declare under penalty of perjury that the foregoing is true and correct. This declaration is executed on October 28, 2021.



— Dr. Ian Foster —

EXHIBIT F

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

NVIDIA CORPORATION
Petitioner

v.

ADVANCED CLUSTER SYSTEMS, INC.
Patent Owner

Patent No. 10,333,768

**DECLARATION OF HENRY TUFO, Ph.D., UNDER 37 C.F.R. § 1.68 IN
SUPPORT OF PETITION FOR *INTER PARTES* REVIEW OF U.S.
PATENT NO. 10,333,768**

TABLE OF CONTENTS

I.	Introduction.....	1
II.	Background and Qualifications	2
III.	Materials Considered for This Declaration	6
IV.	Summary of Opinions.....	7
V.	Understanding of the Law	8
VI.	Technology Background.....	11
VII.	The '768 Patent and Prosecution History	15
	A. Technical Overview of the Patent	15
	B. Prosecution History	22
VIII.	Level of Ordinary Skill in the Art	22
IX.	Claim Construction.....	24
X.	Detailed Analysis.....	24
	A. Ground 1: Schreiner1 in View of Schreiner2, Schreiner3, the Distributed Maple Code, the Maple Guide, the SPARC IV Article, and the AMD Article Renders Obvious Claims 1, 4-10, 18-22, 24-25, 30-31, and 33-34.....	24
	1. Motivations to Combine	25
	2. Overview of the Distributed Maple Publications	29
	3. Claim 1:.....	35
	a. “A computer cluster comprising:”	35
	b. “a plurality of nodes, wherein each of the plurality of nodes comprises a hardware processor,”	35

- c. “wherein one or more of the nodes are configured to receive a command to start a cluster initialization process for the computer cluster, and” 37
- d. “wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, is capable of causing the hardware processor to evaluate mathematical expressions; and” 38
- e. “a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture;” 40
- f. “wherein the plurality of nodes comprises: a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and” 45
 - i. “a first node comprising a first hardware processor” 45
 - ii. “configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel” 45
 - iii. “the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution” 46
- g. “a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of the first mathematical expression evaluation to a third node;” 51

- i. “plurality of processing cores” 51
 - ii. “the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of the first mathematical expression evaluation to a third node” 53
- h. “wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;” 56
 - i. “plurality of processing cores” 56
 - ii. “the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node” 56
- i. “wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface;” 60
- j. “wherein one or more of the nodes are configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to one or more single-node kernels.” 61

i.	“accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other”	62
ii.	“after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to one or more single-node kernels.”	63
4.	Claim 4: “The computer cluster of claim 1, wherein each of the nodes comprises one or more cluster node modules.”	64
5.	Claim 5: “The computer cluster of claim 1, wherein each single-node kernel is stored in a non-transitory computer-readable medium and configured to accept and execute a request.”	68
6.	Claim 6: “The computer cluster of claim 5, wherein each of the nodes comprises one or more cluster node modules, wherein each of the cluster node modules comprises instructions stored in a non-transitory computer-readable medium, and wherein the instructions, when executed by the hardware processor, cause the cluster node module to communicate with the single-node kernel and with one or more other cluster node modules.”	69
7.	Claim 7: “The computer cluster of claim 6, wherein the plurality of cluster node modules act as a cluster.”	70
8.	Claim 8: “The computer cluster of claim 6, wherein the plurality of cluster node modules communicate with one another to act as a cluster.”	70
9.	Claim 9: “The computer cluster of claim 8, wherein the computer cluster includes the user interface.”	70

10. Claim 10: “The computer cluster of claim 9, wherein each cluster node module accepts instructions from the user interface and interprets one or more of the instructions.”71
11. Claim 18: “The computer cluster of claim 1, wherein one or more of the nodes are configured to communicate at least some of the user instructions.”73
12. Claim 19: “The computer cluster of claim 18, wherein one or more of the nodes are configured to communicate at least some of the user instructions to one or more single-node kernels.”73
13. Claim 20: “The computer cluster of claim 1, wherein one or more of the nodes are configured to accept user instructions via one or more of the nodes.”73
14. Claim 21: “The computer cluster of claim 20, wherein one or more of the nodes are configured to communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other.”74
15. Claim 22: “The computer cluster of claim 1, wherein one or more of the nodes are configured to transmit at least some of the user instructions that originate from a user interface.”74
16. Claim 24: “The computer cluster of claim 1, wherein one or more of the nodes are configured to accept user instructions before communicating at least some of the user instructions to one or more single-node kernels.”74
17. Claim 25: “The computer cluster of claim 1, wherein the plurality of nodes are configured to communicate with one another to interpret and translate commands for execution by a plurality of single-node kernels.”75

18.	Claim 30: “The computer cluster of claim 4, wherein each of the plurality of nodes implements asynchronous calls that enable the single-node kernel to perform computation tasks while the cluster node modules are simultaneously communicating with one another.”	77
19.	Claim 31: “The computer cluster of claim 4, wherein intercommunication among the plurality of single-node kernels during thread execution is enabled by the plurality of cluster node modules, and wherein the computer cluster is configured to permit exchange of information between nodes during the course of a parallel computation.”	82
20.	Claim 33: “The computer cluster of claim 1, wherein the plurality of nodes are configured to permit exchange of information between nodes during the course of parallel computation.”	84
21.	Claim 34: “The computer cluster of claim 1, wherein each of the plurality of nodes comprises instructions executable by the hardware processor and configured to implement asynchronous behavior, wherein the instructions comprise: a first instruction to asynchronously send a payload to another node; a second instruction to asynchronously receive a payload from another node; and a third instruction to search for a payload matching a message specifier.”	85
B.	Ground 2: The References in Ground 1 in Further View of the MPI Standard Render Obvious Claims 30, 31, and 34.	89
1.	Claim 30: “The computer cluster of claim 4, wherein each of the plurality of nodes implements asynchronous calls that enable the single-node kernel to perform computation tasks while the cluster node modules are simultaneously communicating with one another.”	89

2.	Claim 31: “The computer cluster of claim 4, wherein intercommunication among the plurality of single-node kernels during thread execution is enabled by the plurality of cluster node modules, and wherein the computer cluster is configured to permit exchange of information between nodes during the course of a parallel computation.”	91
3.	Claim 34: “The computer cluster of claim 1, wherein each of the plurality of nodes comprises instructions executable by the hardware processor and configured to implement asynchronous behavior, wherein the instructions comprise: a first instruction to asynchronously send a payload to another node; a second instruction to asynchronously receive a payload from another node; and a third instruction to search for a payload matching a message specifier.”	92
XI.	Conclusion	93

IX. CLAIM CONSTRUCTION

42. I have been advised and it is my understanding that patent claims in an IPR are governed by the “*Phillips*” approach to claim construction. I have been informed and understand that the claims of a patent must be interpreted in light of the specification—that is, the claim language itself, the written description, and the figures in the patent—as well as the patent’s prosecution history. Claim terms are interpreted based on their ordinary and customary meaning to a POSITA at the time of the invention. As detailed below, I have analyzed the challenged claims under *Phillips* by applying their plain and ordinary meaning as they would have been understood by POSITA at the time of the invention (“plain and ordinary meaning”).

X. DETAILED ANALYSIS

A. **Ground 1: Schreiner1 in View of Schreiner2, Schreiner3, the Distributed Maple Code, the Maple Guide, the SPARC IV Article, and the AMD Article Renders Obvious Claims 1, 4-10, 18-22, 24-25, 30-31, and 33-34.**

43. All elements of claims 1, 4, 7-10, 18-22, 24-25, 30-31, and 33-34 are disclosed in and rendered obvious by Schreiner1 in view of Schreiner2, Schreiner3, the Distributed Maple Code, the Maple Guide, the SPARC IV Article, and the AMD Article.

manipulate information in a symbolic or algebraic manner. ... Maple V maintains and manipulates the underlying symbols and expressions.” EX-1010, 1. In addition, Schreiner1 also discloses using Distributed Maple with a Mathematica kernel, exactly like in the embodiments of the ’289 patent. EX-1008, 306.

e. “a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture;”

71. The claimed mechanism is disclosed in the Distributed Maple Publications as the combination of two software components: `dist.Scheduler`, which “coordinates node interaction,” and `dist.maple`, which “implements the interface between kernel and scheduler.” EX-1008, 310. This scheduler-based message passing mechanism is installed on each node and is used to communicate results of mathematical expression evaluation between nodes using a peer-to-peer architecture: “A session comprises of a set of nodes each of which holds a pair of processes: a kernel and a scheduler.” EX-1008, 319.

72. The annotated figure below from Schreiner1 depicts an example Distributed Maple cluster implementing the claimed mechanism. EX-1008, 1, Fig. 1. The figure shows three nodes and five dots representing additional nodes along with the requisite communication links. The arrows indicate that all nodes are connected to each other, allowing peer-to-peer communications, as expressly taught by Schreiner1:

[A]ll nodes know of each other, i.e. a node knows the address of a machine and the number of a port on which (a thread of) the remote scheduler is listening for connection requests. ***When a node needs to send a message to one of its peers, it can thus establish a direct connection for message transfers.*** The connection remains persistent through the rest of the session such that no more startup overhead is involved.

EX-1008, 315 (emphasis added); *id.*, 317-18;

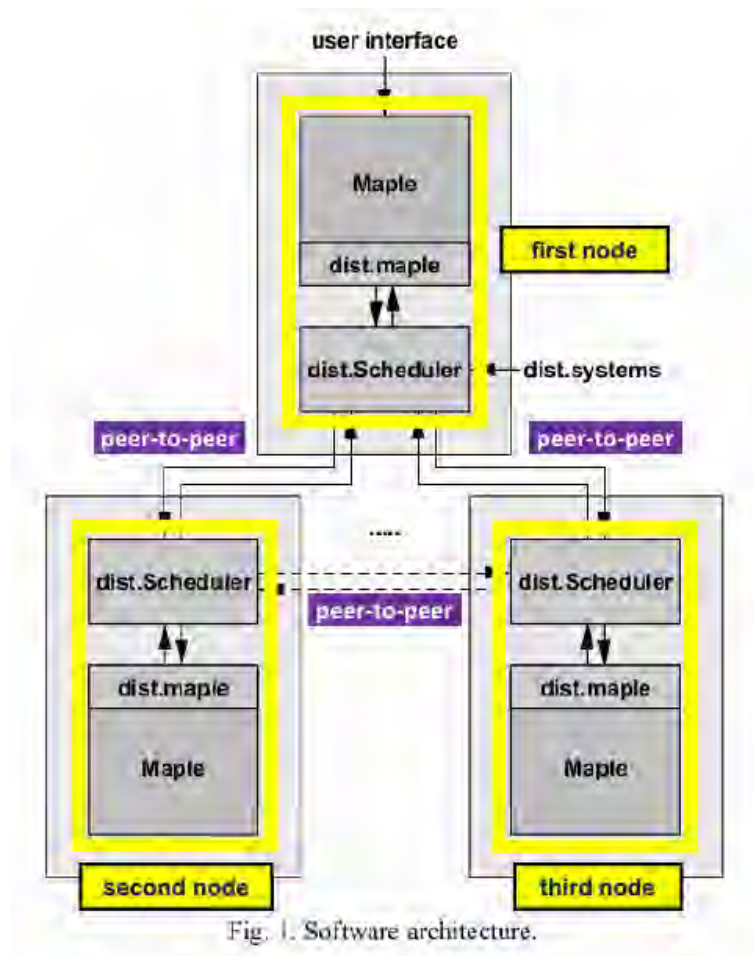


Fig. 1. Software architecture.

EX-1008, 311 (annotated); EX-1009, 23.

73. Schreiner³ further discloses the peer-to-peer message passing mechanism: “When a client wants to send a message to another client through a

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct.

Executed this 2 day of October, 2020.

A handwritten signature in black ink, appearing to read "H. Tufo", written over a horizontal line.

HENRY TUFO, Ph.D.

EXHIBIT G

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

NVIDIA CORPORATION
Petitioner

v.

ADVANCED CLUSTER SYSTEMS, INC.
Patent Owner

Patent No. 10,333,768

**DECLARATION OF HENRY TUFO, Ph.D., UNDER 37 C.F.R. § 1.68 IN
SUPPORT OF PETITION FOR *INTER PARTES* REVIEW OF U.S.
PATENT NO. 10,333,768**

TABLE OF CONTENTS

I.	Introduction.....	1
II.	Background and Qualifications	1
III.	Materials Considered for this Declaration.....	5
IV.	Summary of Opinions.....	9
V.	Understanding of the Law	9
VI.	Technology Background.....	12
VII.	The '768 Patent and Prosecution History	16
	A. Technical Overview of the Patent	16
	B. Prosecution History	23
VIII.	Level of Ordinary Skill in the Art	23
IX.	Claim Construction.....	24
X.	Detailed Analysis.....	25
	A. Ground 1: Schreiner1 in View of Schreiner2, Schreiner3, the Distributed Maple Code, the Maple Guide, the SPARC IV Article, and the AMD Article Renders Obvious Claims 26-27, 29, 35, and 37.	25
	1. Motivations to Combine	25
	2. Overview of the Distributed Maple Publications	29
	3. Claim 26:	35
	a. “A computer cluster comprising:”	35
	b. “a plurality of nodes, wherein one or more of the nodes are configured to receive: a command to start a cluster initialization process for the computer cluster, wherein the cluster initialization process comprises establishing communication	

- among two or more of the nodes; and an instruction from a user interface or a script; and” 35
- i. “a command to start a cluster initialization process for the computer cluster, wherein the cluster initialization process comprises establishing communication among two or more of the nodes” 36
 - ii. “an instruction from a user interface or a script” 37
 - c. “a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using asynchronous calls;” 39
 - d. “wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, is capable of causing a hardware processor to evaluate mathematical expressions;” 46
 - e. “wherein the plurality of nodes comprises: a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and..... 49
 - i. configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, 49
 - ii. the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and 50
 - f. “a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive

- calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of mathematical expression evaluation to a third node;” 55
 - i. “a plurality of processing cores” 55
 - ii. “the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of mathematical expression evaluation to a third node” 58
- g. “wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;” 60
 - i. “a plurality of processing cores” 61
 - ii. the third node is configured to receive the result of mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;” 61
- h. “wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface or the script;” 65
 - i. “wherein one or more of the nodes are configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and after communicating at least some of the user instructions using the mechanism, communicate at

	least some of the user instructions to one or more single-node kernels.”	66
i.	“accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other”	66
ii.	“after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to one or more single-node kernels.”	67
4.	Claim 27: “The computer cluster of claim 26, wherein the asynchronous calls comprise a first command to create a first packet containing: an expression to be sent as payload; and a target node where the expression should be sent; wherein the first command is configured to be called from within a single-node kernel; wherein the single-node kernel is configured to send the first packet to a local cluster node module; and wherein the local cluster node module is configured to forward the expression to the target node.”	69
5.	Claim 29	73
6.	Claim 35	73
a.	“A computer cluster node for evaluating expressions in parallel with other computer cluster nodes, the computer cluster node comprising:”	73
b.	“a hardware processor configured to access one or more non-transitory memory devices comprising program code for a single-node kernel that, when executed, causes the hardware processor to interpret user instructions, to evaluate mathematical expressions, and to produce results of mathematical expression evaluation, wherein the hardware processor comprises multiple processor cores;”	74

- c. “a user connection interface configured to receive a command to start a cluster initialization process for a computer cluster;” 75
- d. “a mechanism to communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture; and” 77
- e. “program code that, when executed, is capable of causing the hardware processor to: receive calls from a second node comprising a second hardware processor configured to access a second memory comprising program code for a user interface and program code for a second single-node kernel, the second single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; execute, using the hardware processor, at least a first mathematical expression evaluation; and communicate a result of the first mathematical expression evaluation to a third node comprising a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the computer cluster node, execute at least a second mathematical expression evaluation using the result of the first mathematical expression evaluation, and communicate a result of the second mathematical expression evaluation to the first node;” 77
 - i. receive calls from a second node comprising a second hardware processor configured to access a second memory comprising program code for a user interface and program code for a second single-node kernel, the second single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution;..... 79

ii.	execute, using the hardware processor, at least a first mathematical expression evaluation; and	79
iii.	communicate a result of the first mathematical expression evaluation to a third node comprising a third hardware processor with a plurality of processing cores,.....	80
iv.	wherein the third node is configured to receive the result of mathematical expression evaluation from the computer cluster node, execute at least a second mathematical expression evaluation using the result of the first mathematical expression evaluation, and communicate a result of the second mathematical expression evaluation to the first node;”	80
f.	“wherein the user connection interface is configured to return at least one result of mathematical expression evaluation to a user interface or a script; and”	81
g.	“wherein the computer cluster node is configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to the single-node kernel.”	81
7.	Claim 37: “The computer cluster node of claim 35, wherein the computer cluster node is configured to permit exchange of information with other computer cluster nodes during the course of parallel computation.”	82
B.	Ground 2: The References in Ground 1 in Further View of the MPI Standard Render Obvious Claims 26-27 and 37	83
1.	Claim 26 (Excerpted Above):	84

2.	Claim 27 (Excerpted Above)	85
3.	Claim 37: “The computer cluster node of claim 35, wherein the computer cluster node is configured to permit exchange of information with other computer cluster nodes during the course of parallel computation.”	86
C.	Ground 3: The References in Ground 1 in Further View of the Maple Reference and Howard Render Obvious Claims 36 and 39.	86
1.	Background on Fourier Transforms.....	86
2.	Claim 36: “The computer cluster node of claim 35, wherein the one or more non-transitory memory devices comprise program code for performing a parallel fast Fourier transform, wherein the program code causes the hardware processor to evaluate a command to perform a Fourier transform on an array comprising a first data portion that is stored on the computer cluster node and a second data portion that is not stored on the computer cluster node.”	91
a.	The Teachings of Howard	92
b.	The Combination of Howard with the References of Ground 1 and the Maple Reference	93
3.	Claim 39: “The computer cluster node of claim 35, wherein the hardware processor comprises a special purpose microprocessor.”	101
D.	Ground 4: The References in Ground 3 in Further View of the FFTW Manual Render Obvious Claim 36.	102
1.	Claim 36 (Excerpted Above)	102
E.	Ground 5: The References in Ground 1 in Further View of PC Magazine 1 and PC Magazine 2 Render Obvious Claim 39.	104
1.	Claim 39: “The computer cluster node of claim 35, wherein the hardware processor comprises a special purpose microprocessor.”	104

F.	Ground 6: The References in Ground 2 in Further View of Nayak Render Obvious Claim 39.....	105
1.	Claim 39 (Excerpted Above)	105
XI.	Conclusion	107

the technology, the type of problems encountered in the art, and the prior art solutions to those problems.

39. I have been advised that a person of ordinary skill in the art, which I will refer to as a POSITA, must be able to make and use the claimed invention based on the information disclosed in the specification of the patent. Accordingly, where information is required to make and use the claimed invention but has not been expressly disclosed in the specification, the patentee effectively acknowledges that such information should be considered to be within the knowledge of the hypothetical POSITA.

40. Based on these considerations, my knowledge and experience, and my evaluation of the scope and content of the '768 patent, a POSITA at the time of the earliest effective filing date of the '768 Patent would have had a Bachelor's degree in computer science, electrical engineering, or an equivalent field, and two years of academic or industry experience in parallel and/or distributed computing.

41. Based on my educational background and experience, I more than satisfied the criteria for a POSITA at the time of the earliest effective filing date of the '768 patent.

IX. CLAIM CONSTRUCTION

42. I have been advised and it is my understanding that patent claims in an IPR are governed by the "*Phillips*" approach to claim construction. I have been

informed and understand that the claims of a patent must be interpreted in light of the specification—that is, the claim language itself, the written description, and the figures in the patent—as well as the patent’s prosecution history. Claim terms are interpreted based on their ordinary and customary meaning to a POSITA at the time of the invention. As detailed below, I have analyzed the challenged claims under *Phillips* by applying their plain and ordinary meaning as they would have been understood by POSITA at the time of the invention (“plain and ordinary meaning”).

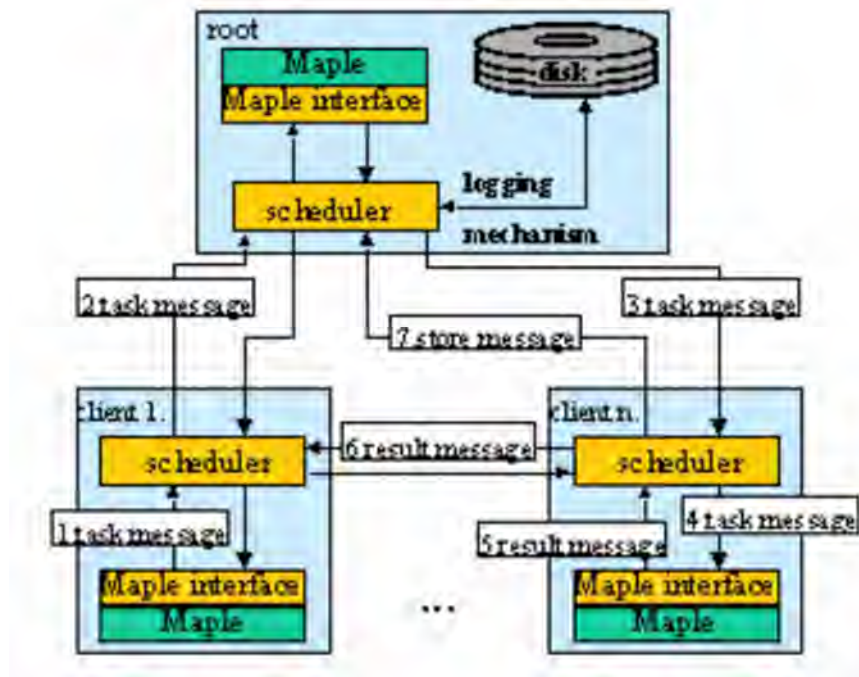
X. DETAILED ANALYSIS

A. Ground 1: Schreiner1 in View of Schreiner2, Schreiner3, the Distributed Maple Code, the Maple Guide, the SPARC IV Article, and the AMD Article Renders Obvious Claims 26-27, 29, 35, and 37.

43. All elements of claims 26-27, 29, 35, and 37 are disclosed in and rendered obvious by Schreiner1 in view of Schreiner2, Schreiner3, the Distributed Maple Code, the Maple Guide, the SPARC IV Article, and the AMD Article.

1. Motivations to Combine

44. This Ground is based on publications related to the same project, called “Distributed Maple,” led by Professor Schreiner at Johannes Kepler University in Linz, Austria. The project resulted in the development of the Distributed Maple system “as a work platform for parallel and networked environments.” EX-1008, 305.



EX-1010, 5.

- d. “a mechanism to communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture; and”

135. The Distributed Maple Publications disclosed a mechanism to communicate results of evaluation with other computer cluster nodes. *See* section X.A.3.c. The nodes communicate those results using a peer-to-peer architecture.

See section X.A.3.c.

- e. “program code that, when executed, is capable of causing the hardware processor to: receive calls from a second node comprising a second hardware processor configured to access a second memory comprising program code for a user interface and program code for a second single-node kernel, the second single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; execute, using

- iii. communicate a result of the first mathematical expression evaluation to a third node comprising a third hardware processor with a plurality of processing cores,**

140. The computer cluster node contains program code, including the scheduler and dist.maple libraries, that communicates results of the first mathematical expression to the “third node” of the claim (e.g., the node in the bottom right of Figure 1 of Schreiner1). *See* sections X.A.3.f and X.A.3.g. The third node comprises a multi-core processor. *See* section X.A.3.f.i.

- iv. wherein the third node is configured to receive the result of mathematical expression evaluation from the computer cluster node, execute at least a second mathematical expression evaluation using the result of the first mathematical expression evaluation, and communicate a result of the second mathematical expression evaluation to the first node;”**

141. The term “the first node” lacks antecedent basis and likely is a drafting error. There are only two possible ways to construe “the first node.”

142. If the “first node” is construed as “the second node” (i.e., the root node in Schreiner1), then the “third node” of the claim (e.g., the node in the bottom right of Figure 1 of Schreiner1) is configured to perform this limitation for the reasons already described for claim 26. *See* section X.A.3.g.

143. If the “first node” is construed as “the computer cluster node,” then it would have been obvious to perform a sequence of tasks that result in the third

node of the claim receiving a result from the computer cluster node, performing another mathematical expression evaluation, and providing the result of that evaluation to the computer cluster node. This is because “[t]he execution of a task may take place on any machine connected to the distributed session,” and “[t]asks may freely create other tasks.” EX-1008, 312. Accordingly, in the normal course of Distributed Maple cluster operation, any node, including the third node, may receive results from any node, and may send results to any node.

- f. “wherein the user connection interface is configured to return at least one result of mathematical expression evaluation to a user interface or a script; and”**


144. The user connection interface of the root node is configured to return a result of the mathematical expression evaluation to the user interface. *See* section X.A.3.h.

- g. “wherein the computer cluster node is configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to the single-node kernel.”**

145. All nodes, including the computer cluster node of this claim, are configured to perform this element, for the same reasons discussed above in section X.A.3.i. As established above, any node, including the claimed node, may

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct.

Executed this 9 day of October 2020.

A handwritten signature in black ink, appearing to read "H. Tufo", written over a horizontal line.

HENRY TUFO, Ph.D.

EXHIBIT H

**IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE**

ADVANCED CLUSTER SYSTEMS, INC.,

Plaintiff,

v.

NVIDIA CORPORATION, NVIDIA
SINGAPORE PTE LTD., NVIDIA
INTERNATIONAL, INC.

Defendants.

CASE NO. 1:19-cv-2032-MN-CJB

DEFENDANTS' FINAL INVALIDITY CONTENTIONS

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

TABLE OF CONTENTS

I.	Introduction.....	1
II.	Final Invalidity Contentions Pursuant To Paragraph 7(C).....	6
III.	U.S. Patent No. 10,333,768	12
A.	Identification of Prior Art	12
1.	Patents and Patent Publications	12
2.	Printed Publications	12
3.	Prior Art Products and Systems	12
B.	Anticipation.....	13
1.	Prior Art Products or Systems.....	13
C.	Obviousness	15
D.	State of the Art of the '768 Patent	16
E.	Obviousness of the Asserted Claims.....	18
1.	Level of Ordinary Skill in the Art.....	18
2.	Obviousness In View of Known Technology	18
i.	Peer-to-Peer Communications in Computer Clusters	21
ii.	Kernels	37
iii.	Claimed Sequence of Three Node Operations.....	41
iv.	<i>Fourier</i> Transforms and Other Computational Libraries.....	60
v.	Special Purpose and Multicore Processors	68
3.	Obviousness in View of Admitted Prior Art and Infringement Allegations	73
F.	Secondary Considerations of Non-Obviousness.....	74
G.	Priority Date and Lack of Written Description.....	80
1.	Claimed Conception Dates	80
2.	May 4, 2007, filing date.....	82
3.	February 14, 2014 filing date.....	86
4.	Applicable Law	87
H.	Invalidity under § 112.....	87
1.	Legal Standard	87
2.	The asserted claims are invalid for lack of written description and for impermissibly adding new matter due to amendments made during prosecution.	88
3.	The asserted claims are invalid for insufficient written description and non-enablement.	104
4.	Indefiniteness	122
I.	Lack of Inventorship	123

DEFENDANTS’ FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

1.	Well-Known Subject Matter	124
2.	Victor Decyk	124
i.	MacMPI	125
ii.	Inter-Node Communication Pattern	135
3.	Theo Gray and Rob Raguet-Schofield	136
4.	Bedros Afeyan, Kirk Won, and/or Vlad Savanchenko	136
5.	Correction of Inventorship	138
J.	Invalidity under § 101	139
IV.	Document Production Accompanying Final Invalidity Contentions	142
A.	Paragraph 8(a) of Scheduling Order (“A copy or sample of the prior art identified . . . that does not appear in the file history of the patent(s) at issue”)	142
B.	Paragraph 8(b) of Scheduling Order (“All agreements that the party opposing infringement contends are comparable to a license that would result from a hypothetical reasonable royalty negotiation”)	143
C.	Paragraph 8(c) of Scheduling Order (“Documents sufficient to show the sales, revenue, cost, and profits for Accused Instrumentalities identified pursuant to paragraph 3(b) of this Order for any period of alleged infringement”)	144
D.	Paragraph 8(d) of Scheduling Order (“All agreements that may be used to support the damages case of the party that is denying infringement.”)	145

DEFENDANTS' FINAL INVALIDITY DISCLOSURES*Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)*

FFT: Our Fourier transform application performs a 2D Cooley-Tukey fast Fourier transform (FFT) [1965] on a 4 channel 1024 by 1024 complex signal. The fast Fourier transform algorithm is important in many graphical applications, such as post-processing of images in the framebuffer, as well as scientific applications such as the SETI@home project [Sullivan et al. 1997]. Our implementation uses three kernels: a horizontal and vertical 1D FFT, each called 10 times, and a bit reversal kernel called once. The horizontal and vertical FFT kernels each perform 5 floating-point operations per output value. The total floating point operations performed, based on the benchFFT [Frigo and Johnson 2003] project, is equal to $5 \cdot w \cdot h \cdot channels \cdot \log_2(w \cdot h)$. To benchmark Brook against a competitive GPU algorithm, we compare our results with the custom OpenGL implementation available from ATI at [ATI 2004a]. To compare against the CPU, we benchmark the heavily optimized FFTW-3 software library compiled with the Intel C++ compiler [INTEL 2003].

Brook for GPU, 2004 at 782.

v. Special Purpose and Multicore Processors

ACS has asserted certain claims, *e.g.*, '768 patent claim 1, that recite processors with a plurality of processing cores. ACS has also asserted claims, *e.g.*, '768 patent claim 39, that recite a “special purpose microprocessor.” With regard to multi-core processor claims, the patents in suit acknowledge that this was conventional and not an inventive feature. *See e.g.*, '768 patent, 7:10-32. The specification recites prior art Pentium and Core Duo processors as examples. *Id.* With regard to the “special purpose microprocessor,” the only example recited by the specification is a DSP digital signal processor. *Id.* DSPs were likewise conventional and not an inventive feature. Notably, the specification lacks any disclosure of a “special purpose processor” with a plurality of processing cores.

Processors with multiple cores were conventional and common well before the alleged inventions by ACS. Such processors were widely used in computing in general, and cluster

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

computing specifically. As just one example, the Matlab *P parallel cluster system and publications disclose the use of dual core processors:

A. Test Platform

- Beowulf cluster with 9 nodes (2 nodes are used in the tests).
- Dual-processor nodes, each with two 1.533GHz Athlon MP processors.
- 1GB DDR RAM on each node. No swapping occurred during benchmarks.
- Fast ethernet (100Mbps/sec) interconnect. Intel Etherfast 410T switch.
- Linux 2.4.18-4smp

Choy 1 at 4.

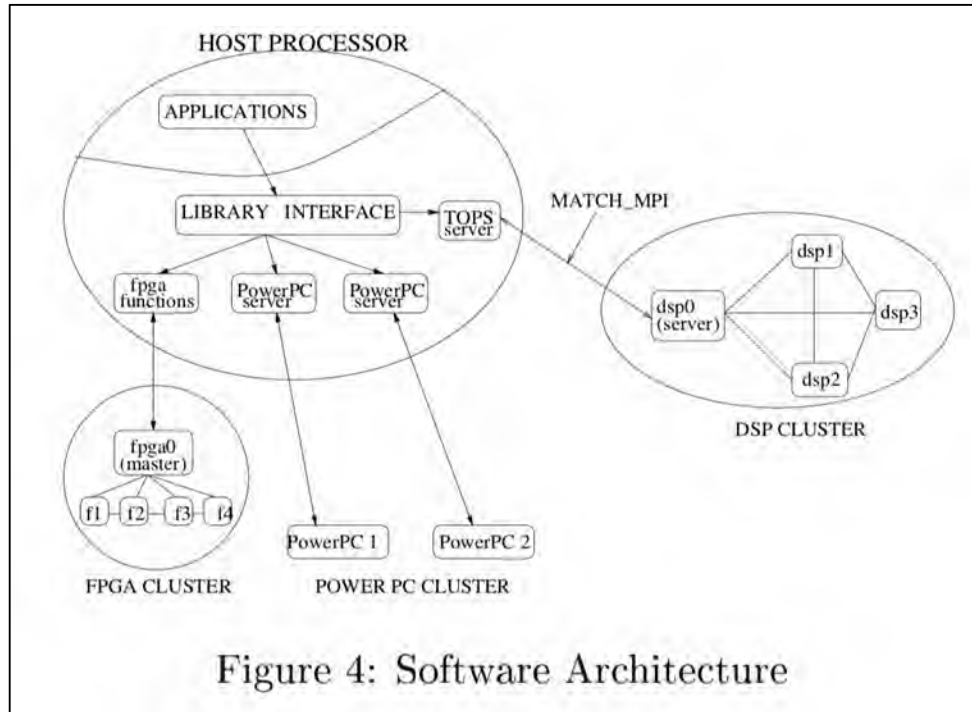
Likewise, DSPs were known to be advantageous in cluster computing for certain types of problems. As one example, Nayak teaches the advantages of cluster computing on DSPs:

The DSP architecture has some custom hardware for multiply and accumulate type of operations, which make them ideal for signal and image processing applications in comparison to the general purpose processors. Hence, it is wise to implement computation intensive matrix operations on the DSP as they would take a longer time on general purpose processors.

Nayak at 4.

Nayak teaches a cluster with a combination of DSP and/or FPGA nodes, in addition to other processor nodes. It uses a Matlab kernel and MPI for peer-to-peer message passing, making it obvious to combine with Matlab *P and other references using MPI and similar peer-to-peer protocols, such as PVM or Distributed Maple's scheduler messaging mechanism:

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)



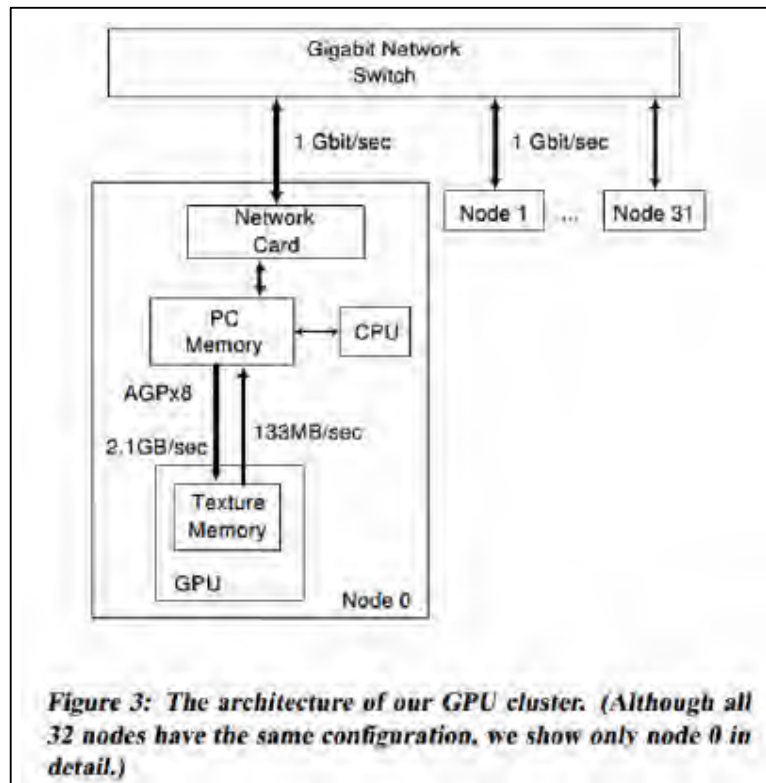
Nayak, Fig. 4.

As a further example of knowledge in the art regarding multiple core and special purpose processors, U.S. Pat. No. 7,996,592 teaches: “When a particular application/project/job requires more processing power than a single processor is capable of providing, it becomes necessary to provide a coprocessor, such as a digital signal processor (DSP) or a floating point unit (FPU).” ’592 patent, 1:10-15. Accordingly, the use of special purpose processors was conventional, and obvious whenever extra computational power was desired.

Moreover, the Stony Brook cluster was implemented and publicly known and used at Stony Brook University as early as 2003. It comprised over 30 nodes, each node comprising a CPU with at least a plurality of processing cores and an NVIDIA GPU: “We have built a cluster with 32 computation nodes connected by a 1 Gigabit Ethernet switch. Each node consists of a dual-CPU HP PC with an nVIDIA GeForce FX 5800 Ultra.” Fan1 at Our GPU Cluster. Each node was

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

interconnected with each other node with a peer-to-peer communication mechanism comprising a high-speed network for exchanging MPI protocol messages between the nodes:

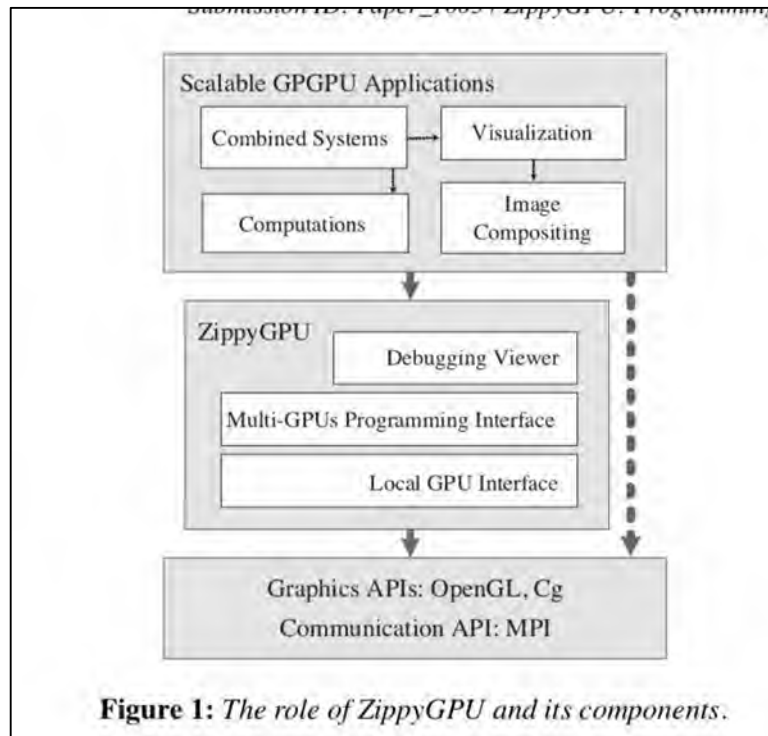


Fan1 at Fig. 3; Fan1 at Our GPU Cluster.



Fan1 at Fig. 2.

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)



Fan 6 at 3.

ACS has asserted that its claims cover GPU-based parallel computing. While this is incorrect, and the '768 patent specification does not enable or disclose GPU parallel processing, the use of GPUs for parallel processing in computer clusters with peer-to-peer communications was known in the art well before the alleged inventions of the '768 patent. To the extent ACS contends that the '768 patent enables and describes the application of its claims to GPU-based parallel processing, it would have been obvious to combine the teachings of any of the computer cluster systems and references in these contentions with the Stony Brook cluster as described by Fan et al. For example, the Stony Brook cluster implemented the MPI messaging protocol, making it obvious to combine with other MPI-based systems and references such as Star*P or MultiMATLAB, or with references with similar protocols, such as PVMaple or Distributed Maple.

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

3. The asserted claims are invalid for insufficient written description and non-enablement.

NVIDIA contends that every asserted claim is invalid for lack of a written description because the specification fails to “demonstrate that the patentee possessed the full scope of the invention recited in” the claims, at least as interpreted by ACS. *LizardTech*, 424 F.3d at 1345. NVIDIA also contends that every asserted claims is invalid for non-enablement for failure to “enable the full scope of the claims.” *Automotive Technologies Intern. Inc. v. BMW of North America, Inc.*, 501 F.3d 1274, 1285 (Fed. Cir. 2007); *Genentech, Inc. v. Novo Nordisk A/S*, 108 F.3d 1361, 1365 (Fed. Cir. 1997); *id.* at 1366 (even though the “specification need not disclose what is well-known in the art,” “[t]ossing out the mere germ of an idea does not constitute enabling disclosure”); *In re Wright*, 999 F.2d 1557, 1561 (Fed. Cir. 1993); *In re Wands*, 858 F.2d 731, 737 (Fed. Cir. 1988) (*Wands* factors).

The '768 patent is directed to retrofitting mathematical programs like Mathematica to add the capacity for parallel computations: “The present disclosure relates to the field of cluster computing generally and to systems and methods for ***adding cluster computing functionality to a computer program***, in particular.” '768 patent, 1:14-17, 2:4-6 (emphasis added).

Figs. 1 and 2 provide an overview of the hardware and software of a cluster implementing the '768 patent's claimed invention. *Id.*, 4:36-6:37, Figs. 1, 2.

Fig. 1 shows the hardware components, including multiple computer systems (computer systems 110, 120, and 13), processors (processor 112a, 112b, 122a, 122b, 132), memories (memories 114, 124, 134), and optional storage (storage 116, 126, 136). *Id.*, 4:36-5:15, Fig. 1.

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

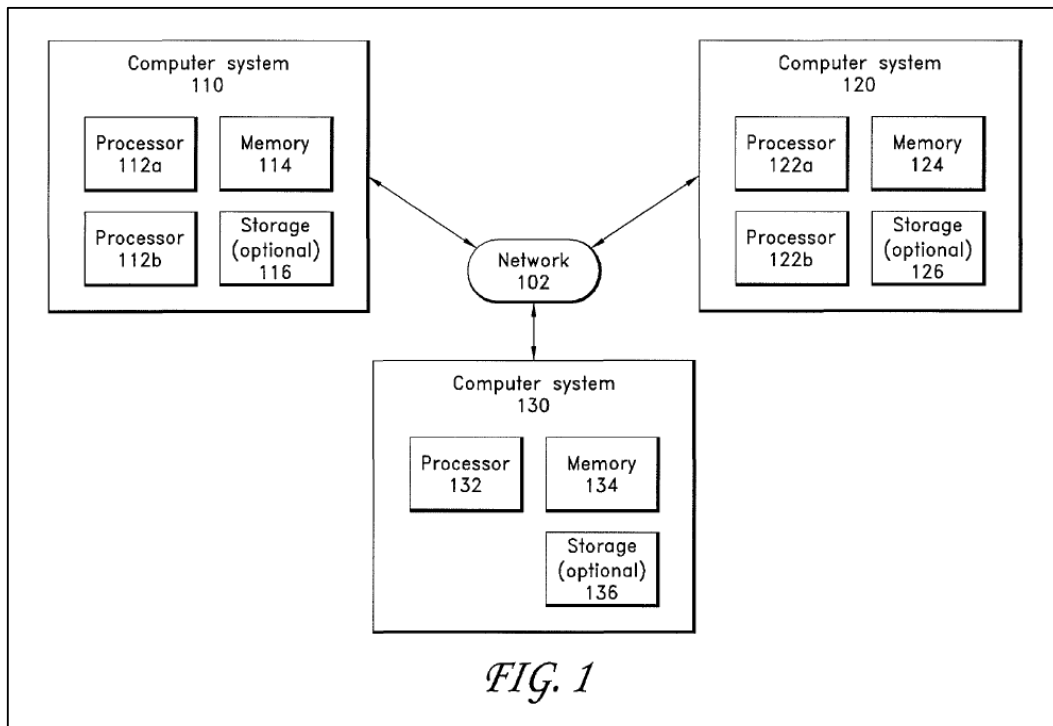
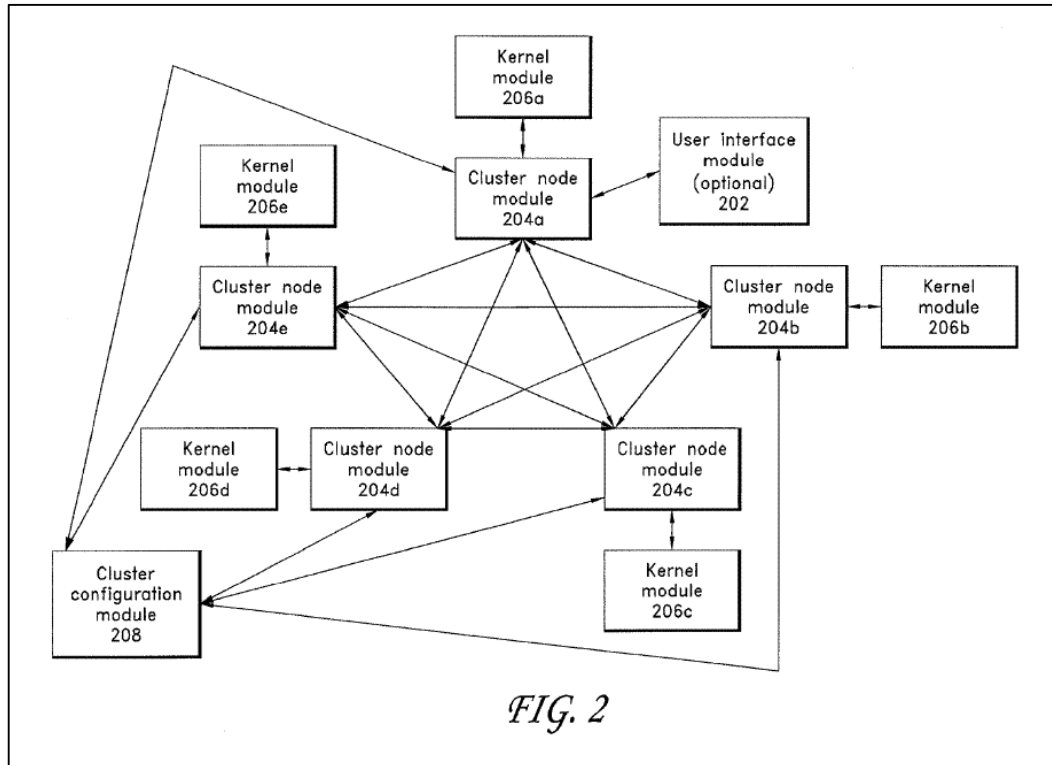


Fig. 2 shows the software components and their relationships to each other. *Id.*, 5:16-6:37, Fig. 2. The kernel modules (206a-e) and user interface module (202) are the existing software programs that the claimed invention is seeking to augment. The cluster node modules (204a-e) and cluster configuration module (208) include new software added by the claimed invention. The cluster node modules add parallel computing capabilities despite using modules designed for only single-threaded work: “The cluster node modules 204a-e provide MPI calls and/or advanced cluster functions that implement cluster computing capability for the single-threaded kernel modules.” *Id.*, 5:61-64. The cluster node modules can be inserted into the existing single-node programs, because they “are configured to look and behave like a kernel module 206a from the perspective of the user interface module 202” and “are configured to look and behave like a user interface module 202 from the perspective of a kernel module 206a.” *Id.*, 5:61-6:2.

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)



Thus, the solution disclosed in the '768 patent is narrow and limited. The solution is directed only at retrofitting cluster computing onto existing single-node kernel programs. For communication, it adds only an MPI module and two queues. For parallel calculations, it adds a limited “advanced functions module” with a toolkit of parallelized functions.

Despite this narrow solution, ACS is trying to extend its claims to cover “distinctly different” embodiments that the '768 patent does not describe or enable. *Auto. Tech. Int'l v. BMW of N. Am.*, 501 F.3d 1274, 1285 (Fed. Cir. 2007). To the extent ACS is successful in expanding the claim scope to cover those embodiments, its claims are invalid. *Liebel-Flarsheim v. Medrad*, 481 F.3d 1371, 1380 (Fed. Cir. 2007) (“The irony of this situation is that Liebel successfully pressed to have its claims include a jacketless system, but, having won that battle, it then had to show that such a claim was fully enabled, a challenge it could not meet. The motto, ‘beware of what one asks for,’ might be applicable here.”).

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

The asserted claims are lack sufficient written description and are not enabled for at least the following reasons.

First, each asserted independent claim refers generally to hardware “processor[s]” and “processing cores,” but the ’768 patent does not describe or enable the use of its claimed invention with respect to all types of hardware. Instead, the ’768 patent refers only to CPUs (Central Processing Units) and DSPs (Digital Signal Processors). Nothing in the ’768 patent either describes or enables how to use its alleged claimed invention with GPUs (Graphics Processing Units). The ’768 patent also fails to enable the application of its invention to DSPs, mentioning them only by name without any disclosure of how a person of skill could make and use a cluster computer with DSPs.

The following provides a list of generic processor language used in the asserted independent claims:

Claim 1: “a hardware processor,” “the hardware processor,” “a first hardware processor,” “a second hardware processor with a plurality of processing cores,” “a third hardware processor with a plurality of processing cores”

Claim 26: “a hardware processor,” “a first hardware processor,” “a second hardware processor with a plurality of processing cores,” “a third hardware processor with a plurality of processing cores”

Claim 29: “a hardware processor,” “a first hardware processor,” “a second hardware processor with a plurality of processing cores,” “a third hardware processor with a plurality of processing cores”

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

Claim 35: “a hardware processor,” “the hardware processor,” “the hardware processor comprises multiple processor cores,” “a second hardware processor,” “a third hardware processor with a plurality of processing cores”

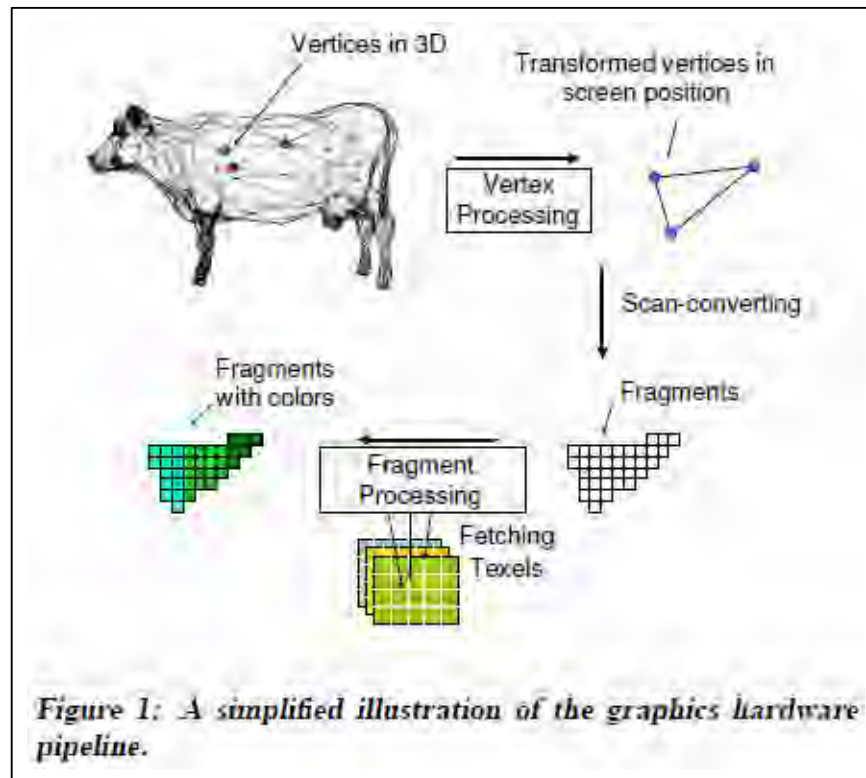
The only processors listed in the patent are CPUs and DSPs:

In one embodiment, the computer system **110** includes one or more processors **112a-b**. The processors **112a-b** can be one or more general purpose single-core or multi-core microprocessors such as, for example, a Pentium® processor, a Pentium® II processor, a Pentium® Pro processor, a Pentium® III processor, Pentium® 4 processor, a Core Duo® processor, a Core 2 Duo® processor, a Xeon® processor, an Itanium® processor, a Pentium® M processor, an x86 processor, an Athlon® processor, an 8051 processor, a MIPS® processor, a PowerPC® processor, an ALPHA® processor, etc. In addition, one or more of the processors **112a-b** can be a special purpose microprocessor such as a digital signal processor. The total number of processing

'768 patent, 7:11-23; *id.*, 8:29-40, 9:48-59.

GPUs are fundamentally different processors from CPUs. CPUs are intended as general-purpose processors that act as the “brain” of a computer system. In contrast, at the time of the alleged invention of the '768 patent, GPUs were designed to accelerate graphics applications. The following figure shows an example of the type of operations in the “graphics hardware pipeline” that GPUs were designed to perform:

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)



Fan et al., (“Fan 1”) GPU Cluster for High Performance Computing (2004).

The differences between CPUs and GPUs are vast. Whereas CPUs can operate independently from GPUs and other processors, GPUs are dependent upon, and require a CPU, to operate. No system can include only a GPU. Whereas CPUs are designed for low-latency, sequential code performance for only a few threads, GPUs are designed for rapidly switched, massively data parallel operations for many threads, requiring a fundamentally different memory and registry architecture. Whereas CPUs employ up to a small number of general-purpose processing cores in, *e.g.*, a “multi-core” CPU, GPUs contains hundreds or thousands of special-purpose cores, such as CUDA cores, tensor cores, and ray tracing cores in, *e.g.*, a “many-core” GPU.

The differences between GPUs and CPUs also extend to the concept of the recited computer cluster nodes. A node in a computer cluster is recognized in the art as comprising a single

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

instance of an operating system running on a processor. This is reflected by the '768 patent, which teaches that each cluster node in Fig. 1 is a computer system with an operating system, e.g., a computer system 110 or 120. '768 patent, 7:1-9, 7:59-63. The specification further acknowledges that setting up a computer cluster requires each node to have "an operating system with discoverable network services." '768 patent, 22:26-35. This association of a node in a computer cluster with the conventional computer – a CPU running an operating system – is directly contrary to how a GPU operates. A GPU is not a computer or a processor running an operating system instance. Rather, it is a device within a computer that accelerates the computer's computations under certain workloads. In this regard, several GPUs in a single computer, all operating to accelerate computations of a CPU running an operating system, are not separate nodes in a cluster of computers. Instead, the GPUs are all within a single computer. The '768 specification does not disclose or enable any embodiment of a computer cluster comprising a single computer, regardless of whether some of the alleged nodes are GPUs or CPUs. Certainly, there is no disclosure or enablement of a computer cluster with nodes comprising GPUs. The fundamentally different hardware architecture also results in fundamentally different software requirements. GPUs use a much different software stack from a CPU, and thus much different from anything described or enabled in the '768 patent. For example, the accused NVIDIA GPUs use a SIMT (single instruction multiple thread) software architecture, employing threads, warps, and blocks. NVIDIA also provides a parallel-processing API called CUDA (Compute Unified Device Architecture) that uses streams, CUDA kernels, thread and block indices, shared memories, barrier synchronization, CUDA graphs, heterogenous programming, and a wide variety of other software architecture features that are absent from the CPU software discussed in the '768 patent.

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

At the time of the '768 patent's earliest cited priority date, GPUs were programmed using graphics-specific APIs (Application Programming Interfaces) and were required to use graphics-specific pipeline hardware. These APIs allowed developers to create programs that would use the GPUs to create advanced graphics for video games, as an example. But the requirement to program GPUs with graphics APIs made it difficult to program GPUs for general-purpose calculations. They required developers to adapt the graphics pipeline to perform entirely different functions not intended by the graphics APIs. As examples, if a developer wanted to use a GPU for general-purpose calculations, it needed to manually pack data into textures, using shading operations as a workaround for general mathematical operations; general mathematical expressions needed to be expressed in terms of graphics primitives, such as textures and triangles; and needed to create its own workaround for basic computing functions, such as the gather operation, which were missing from the graphics APIs.

Programming GPUs also required accounting for the limitations of the graphics hardware. These limitations included narrow data structures, such as having only four natively supported formats (float, float2, float3, and float4) rather than allowing for the more-complex user-defined data structures available for CPUs. They also included limitations on shader instruction counts, number of shader outputs, and texture sizes. GPU hardware designs were tailored to the graphics pipeline and graphics APIs, having very restricted memory reading and writing capabilities. For example, GPU programs called shaders did not have the ability to perform scatter operations, which refer to writes to memory with calculated memory addresses. Instead, to write a result to memory required configuring the data as a pixel color value and configuring the frame buffer to write the result to a two-dimensional frame buffer array. Passing results from one computation

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

step to another also required writing pixels to a frame buffer array and then using that array as a texture map input into a shader program.

The difficulties with programming GPUs limited that field to only the most advanced graphics developers and required significant, custom work for even the most basic programs.

Nothing in the '768 patent describes or enables the difficult adaptation of GPUs to perform those functions. The '768 patent does not mention GPUs, let alone describe or enable how to overcome the difficulties of performing general mathematical operations on a GPU. To the contrary, the '768 patent simply assumes that the claimed "processors" can be programmed simply using the basic tools for CPUs. Indeed, ACS trumpeted the supposed ease of programming for its alleged invention, without providing any solution for how to program GPUs. (*E.g.*, U.S. Provisional 60/850908 at 1 ("Our idea was to combine this tool [Mathematica] with the easy-to-use cluster computing solution embodied by Pooch and MacMPI to enable supercomputing-like behavior and computations within Mathematica.")).

Second, the '768 patent is limited to adding cluster computing functionality to computer programs that were designed to be run on a single node. "Some computer programs can run on only a single node because, for example, they are coded to perform tasks serially or because they are designed to recognize or send instructions to only a single node." *Id.*, 1:26-29; *id.*, 11:30-32 ("The kernel modules 206a-e, which include single-threaded program code, are each associated with one of the processors 112a, 112b, 122a, 122b, 132.")).

The following provides a list of language used in the asserted independent claims requiring "single-node" programs:

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

Claims 1, 26, 29: “each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel,” “a first single-node kernel,” “the first single-node kernel,” “one or more single-node kernels”

Claim 35: “program code for a single-node kernel,” “a second single-node kernel,” “the second single-node kernel,” “the single-node kernel”

Contrary to the limited and narrow disclosure in the '768 patent, and the claim language itself, ACS is seeking to expand the scope of its claims to cover systems that do not retrofit single-node programs with cluster computing functionality. Instead, ACS is trying to capture programs that are intrinsically multi-node programs. As noted above, a GPU cannot be operated independently but instead requires a CPU for its operation. An application program intended to be run on both a CPU and one or more GPUs is a multi-node program, not a single-node program.

Regardless of these limitations, in this litigation, ACS is attempting to cover programs that are intended to be run on multiple nodes, including at least a CPU and one or more GPUs. These systems should be outside the properly construed claim scope because they are not “single-node” programs but are instead designed from the ground up to operate on multiple nodes. But if ACS is successful in expanding the claim scope to cover them, the claims are invalid because nothing in the '768 patent describes or enables how to add cluster computing functionality to a program that incorporates both CPUs and one or more GPUs. As described in detail above, CPUs and GPUs are fundamentally different devices and describing and enabling a program to be run on a CPU does not describe or enable a program that will be run across a CPU and one or more GPUs.

Third, the '768 patent is limited to single-node programs that included a kernel: “Some application programs include an interpreter that executes instructions provided to the program by a user, a script, or another source. Such an interpreter is sometimes called a ‘kernel’ because, for

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

example, the interpreter can manage at least some hardware resources of a computer system and/or can manage communications between those resources and software (for example, the provided instructions, which can include a high-level programming language).” *Id.*, 1:29-37; *id.*, 23:9-13 (“A kernel module 206 typically includes program code for interpreting high-level code, commands, and/or instructions supplied by a user or a script into low-level code, such as, for example, machine language or assembly language.”). The ’768 patent describes Mathematica as an example of a program with a single-node kernel. “An example of a software package that includes a kernel that is designed to communicate with a single node is Mathematica from Wolfram Research, Inc. (‘Mathematica.’)” *Id.*, 1:38-42; *id.*, 2:18-21.

The claim language quoted above for the “single-node” limitation also requires that the programs-at-issue include a “kernel.”

Despite the ’768 patent’s narrow disclosure and the claim language, ACS is improperly trying to stretch the reach of its claims to cover programs that do not use an interpreter but instead compile high-level code before execution. ACS’s attempts should be rejected. But if the claims are extended as broadly as ACS is seeking, the claims would be invalid for lack of written description and non-enablement. The ’768 patent relies on the kernel within the software program having the capability of translating high-level instructions into code that could be executed by the processor. The ’768 patent does not describe or enable a pre-compiled system, particularly one that would include both a CPU and one or more GPUs.

As another example, it appears from ACS’s inadequate and confused infringement contentions that ACS has implicated “CUDA kernels” as allegedly practicing the ’768 patent claims’ “kernel” limitations. While “CUDA kernels” share a word with the “kernels” recited in the patents-in-suit, they are entirely different and unrelated concepts. CUDA kernels come from

DEFENDANTS' FINAL INVALIDITY DISCLOSURES***Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)***

the field of stream computing, a different field of art. The CUDA kernel is not an application program, it is not designed to run on a single computer, and it does not evaluate mathematical expressions by interpreting high level code provided by a user or a script into low level code. Instead, CUDA kernels are user-defined C++ functions that, when called, are executed N times in parallel by N different CUDA threads as opposed to only once like regular C++ functions. Moreover, the stream computing concepts relevant to the CUDA kernel and the accused CUDA functionality more generally are not disclosed in the specifications of the patents in suit. The specification does not disclose the alleged inventions' applicability to stream computing generally, or to the accused NVIDIA functionality more specifically.

In addition, CUDA kernels, unlike the claimed kernels, execute only on GPUs. The claimed kernels execute on computer cluster nodes, each such node comprising a computer with a processor under the control of an operating system. There is no teaching in the '768 specification of any kernel that can execute on anything other than a conventional computer with a conventional CPU. Certainly, there is no teaching of how the alleged invention may operate on, or even be relevant to, a stream computing architecture of the type pioneered by the Brook stream computing language, the Merrimac supercomputer, and later adopted by CUDA.

Nor is there a teaching in the '768 specification of any kernel that could operate on a GPU. In fact, none of the kernels discussed in the '768 specification could perform any computations on a GPU, and the '768 specification does not provide any description enabling a POSITA to adapt Mathematica kernels (or other recited kernels, such as Maple) to GPU computation. At the time of the alleged invention such an adaptation would have required extensive reprogramming and was not within the ordinary skill.

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

Fourth, the '768 patent purports to add communication capabilities to programs that would otherwise be limited to single nodes. The '768 patent recognized that the prior art already included systems that tied Mathematica (and other single-node kernel programs) together to achieve distributed computing. "A product known as gridMathematica, also from Wolfram Research, Inc., gives Mathematica the capability to perform a form of grid computing known as 'distributed computing.'" '768 patent, 1:43-46. The '768 patent tried to distinguish its claimed invention from that prior art by asserting that this prior art lacked peer-to-peer communications: "Grid computers include a plurality of nodes that generally do not communicate with one another as peers.... Grid computers include at least one node known as a master node that manages a plurality of slave nodes or computational nodes. In gridMathematica, each of a plurality of kernels runs on a single node. One kernel is designated the master kernel, which handles all input, output, and scheduling of the other kernels (the computational kernels or slave kernels). Computational kernels receive commands and data only from the node running the master kernel." *Id.*, 1:47-59, 6:26-32.

Despite suggesting that its invention is directed to peer-to-peer communication generally, the '768 patent's disclosure is limited to using MPI along with two queues: a receiving message queue and a message receiving queue. *Id.*, 12:47-16:37 (MPI), 21:62-22:15 (received message queue), 22:16-24 (message receiving queue).

The following provides a list of language used in the asserted independent claims with respect to the required communication architecture:

Claim 1: "a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture"

Claim 26: "a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using asynchronous calls"

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

Claim 29: “a mechanism for the nodes to communicate results of mathematical expression evaluation with each other”

Claim 35: “a mechanism to communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture”

At the time of the claimed invention, MPI was a well-established mechanism for transferring data between CPUs. But the '768 patent provides no description or enablement of how to use MPI in transferring data between GPU cores, between GPUs, or between GPUs and CPUs. Nor does the '768 patent describe or enable any way of transferring data between GPU cores, between GPUs, or between GPUs and CPUs. To the contrary, the '768 patent does not mention GPUs at all.

Nor does anything in the '768 patent enable the NVLink communications features that have been accused of infringement. The accused NVLink technology is distinctly different from the off-the-shelf networking described in the patent. Unlike the networking discussed in the patents, NVLink provides a high-speed, direct GPU-to-GPU interconnect. Whereas MPI requires explicit message passing where the nodes do not share memory, with NVLink-connected GPUs, programs can execute directly on memory that is attached to another GPU as well as on local memory.

NVLink is a memory interconnect while MPI is a message passing interface that runs at a completely different and higher level of abstraction. The '768 patent specification does not disclose or enable message passing over a wire-level memory interconnect. To the contrary, the '768 patent teaches that nodes communicate over conventional networking technologies: “Network 102 includes one or more of a LAN, a WAN, a wireless network, an intranet, or the Internet.” '768 patent, 4:63-67. The conventional networks disclosed in the '768 patents enable MPI message passing between different computers in the claimed computer cluster. On the other

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

hand, NVLink is a collection of wires passing low-level packets between device memories in a single computer.

MPI is a communication protocol for passing messages between computers. NVLink, on the other hand, facilitates a unified memory space where devices such as GPUs are generally not aware of whether the data they are accessing is local or remote. This makes the very concept of passing messages between node computers in a computer cluster inapplicable to a memory fabric like NVLink. Instead of moving from one node to another in a message, data is stored at a virtual memory address. Whether that address is local to the device accessing the memory is generally hidden from software running on the device.

Unlike the message passing between separate computers described in the '768 patent, the memory fabric enabled by NVLink allows multiple GPUs to operate as a single GPU. This is fundamentally different than the claimed computer cluster with individual computer nodes communicating messages to each other. There is no disclosure or enablement of the alleged invention operating with a memory fabric. In fact, the concept of node-to-node data communication is inapplicable to a memory fabric like NVLink. Rather than communicating a message containing data addressed to another computer, a memory client places data at a memory address, with all memory addresses within the same computer.

Enablement of a GPU-centric memory interconnect by NVLink required a myriad of inventions by NVIDIA engineers, none of which are disclosed or even hinted at in the '768 patent specification. NVLink uses same address ordering and a non-tree topology to allow multipath transmission. This increases the throughput of each link, and links are aggregated for even faster transmission. Moreover, NVLink is tailored to short-distance transmission paths found within a single server, allowing further maximization of throughput. In addition, NVLink supports GPU-

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

oriented memory operations arising from the graphics-specific nature of GPU architectures. The '768 patent, on the other hand, relies on conventional MPI message passing over conventional networks with long transmission paths and software overhead associated with computer-to-computer communication over generic networks.

Fifth, the asserted claims of the '768 patent are not supported by the written description because of a failure to claim an element that was essential to the invention as described in the specification. *Gentry Gallery, Inc. v. Berkline Corp.*, 134 F.3d 1473 (Fed. Cir. 1998). In the '289, '877 and '612 IPR POPRs, ACS claimed that the “at least two fundamental improvements” described in the specification are (1) that the “inventors interposed a ‘cluster node module’ between the user interface and the kernel;” and (2) that the “inventors designed the cluster node modules to intercommunicate with each other in a peer-to-peer architecture.” '612 POPR at 1, 3. ACS further argued that “the inventors coined the phrase ‘cluster node module’ to encapsulate essential features of the modules that interconnect the nodes.” *Id.* at 15.

Accordingly, ACS admitted that the “cluster node module” is a “fundamental improvement” and an “essential feature” of their inventions, which is recited in the independent claims of the '289, '877 and '612 patents. However, this fundamental and essential feature is omitted from the '768 patent independent claims, such that their scope is broader than the supporting disclosure, invalidating the claims under *Gentry Gallery*.

Moreover, ACS stated that “the specification does not disclose any kernel that accepts instructions from a user interface and forwards instructions to a cluster node module.” '612 POPR at 12. “A POSITA would understand that the interposition of the cluster node modules between the user interface and the kernels – such that the cluster node modules accept instructions from the

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

user interface without those instructions passing through the kernels – is an important feature of the design.” *Id.*

Yet, the '768 patent claims cover an arrangement that ACS has admitted is not disclosed by the specification – one where a kernel accepts instructions from a user interface and forwards instructions to a cluster node module. For example, '768 patent claim 1 recites “a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution.” The other independent claims recite similar limitations.

This claim language does not prevent the kernel from receiving user instructions directly; in fact, that method is a logical implication of the recitation of a “user interface” and a “kernel configured to interpret user instructions and distribute calls.” However, according to ACS, a direct passing of user instructions from the user interface to the kernel, without the interposition of a cluster node module, is contrary to a POSITA’s understanding of the specification: “A POSITA would understand that the interposition of the cluster node modules between the user interface and the kernels – such that the cluster node modules accept instructions from the user interface without those instructions passing through the kernels—is an important feature of the design.” '612 POPR at 12. Accordingly, the '768 patent asserted claims extent coverage to subject matter that ACS concedes is not disclosed by, and is contrary to, the specification’s teachings.

This defect extends to all of the asserted '768 patent claims, including the dependent claims reciting a “cluster node module,” such as claim 4. These claims, when read together with the independent claims, are not supported by the written description at least because their claim

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

limitations do not exclude a “kernel that accepts instructions from a user interface and forwards instructions to a cluster node module.” ’612 POPR at 12.

Finally, the failure of the ’768 patent to describe or enable the claims extends to the following limitations as well. Claim 1 recites “wherein one or more of the nodes are configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to one or more single-node kernels.” This limitation is not supported by written description and is not enabled. Similar limitations are found in claims 26 and 29, and claim 35 recites this limitation in the context of a claim to a node. These claim limitations are not supported by written description or enabled. Claim 35 recites “communicate the result of the second mathematical expression evaluation to the first node.” This limitation is not supported by written description or enabled. Claim 25 recites “the plurality of nodes are configured to communicate with one another to interpret and translate commands,” which is not supported by written description or enabled. Claim 36 recites performing a Fourier transform “on an array comprising a first data portion that is stored on the computer cluster node and a second data portion that is not stored on the computer cluster node.” This limitation is not supported by written description or enabled. Claim 10 recites “wherein each cluster node module accepts instructions from the user interface and interprets one or more of the instructions.” This limitation is not supported by written description or enabled. Claim 39 recites “wherein the hardware processor comprises a special purpose microprocessor.” This limitation is not supported by written description, which does not disclose any multi-core processor (as required by independent claim 35) that is a special purpose microprocessor, and is not enabled.

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

4. Indefiniteness

Under 35 U.S.C. § 112, a “patent is invalid for indefiniteness if its claims, read in light of the specification delineating the patent, and the prosecution history, fail to inform, with reasonable certainty, those skilled in the art about the scope of the invention.” *Nautilus, Inc. v. Biosig Instruments, Inc.*, 134 S. Ct. 2120, 2124 (2014). At least the following elements of the asserted ’768 patent claims are indefinite, because they fail to inform a person of ordinary skill in the art about the scope of the invention with reasonable certainty.

Claim 35 is indefinite for at least the reason that the limitation “communicate a result of the second mathematical expression evaluation to the first node” lacks antecedent basis, and fails to inform a POSITA with reasonable certainty about the scope of the invention.

Claim 25 is indefinite for at least the reason that the limitation “the plurality of nodes are configured to communicate with one another to interpret and translate commands” fails to inform a POSITA with reasonable certainty about the scope of the invention.

Furthermore, each asserted claim is indefinite under pre-AIA 35 U.S.C. Section 112, second paragraph, because the claims recite both an apparatus and a method for using that apparatus. A claim does not meet the requirements of this statute, and is therefore invalid for indefiniteness, when it “recit[es] both an apparatus and a method of using that apparatus.” *IPXL Holdings, L.L.C. v. Amazon.com, Inc.*, 430 F.3d 1377, 1384 (Fed. Cir. 2005). Here, all of the claims recite an apparatus and also one or more configurations or steps that will not occur until the apparatus is used in a specific way.

NVIDIA expressly reserves the right to amend its Final Contentions and pursue invalidity based on additional basis for indefiniteness as the case proceeds and after claim construction occurs, or in view of positions taken by ACS regarding invalidity or infringement.

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

i. MacMPI

According to ACS, the subject matter claimed in the '768 patent is the result of the development of a product now called Supercomputing Engine for Mathematica ("SEM"),⁴ which ACS also claims practices the asserted patent. SEM—and the claimed subject matter of the '768 patent, however, is nothing more than a combination of Mathematica with MacMPI—a product developed by Viktor Decyk. *See, e.g.*, DR_NVIDIA_000468 (Pooch Manual), at _476 ("The first is a communications library named MacMPI, authored by Dr. Viktor K. Decyk. MacMPI provides a means for executables on each Macintosh to communicate with one another. It is a source code wrapper library that translates the calls by the parallel code into calls directly to the MacOS. MPI (Message-Passing Interface) is an industry-standard (the high-performance parallel computing industry, that is) application programming interface supported on almost all large parallel computers (Cray, IBM SP, Fujitsu, SGI, *etc.*). Using the MacOS, MacMPI supports a comm only used subset of those MPI calls."); POLYMATH_0003532 (Dean Dauger referring to MacMPI as "Viktor's MacMPI"); POLYMATH_0006479, at _6483 ("Submitting this problem to Viktor Decyk, he pointed out that there might be a bug in the Mac's Open Transport feature his MPI implementation uses, regarding the size of the messages we sent. Therefore, he provided us a new MPI source (initially written in C) using a different way to pass messages that we tried."); POLYMATH_0013082 ("I am using Pooch which is supported by MPI written by Viktor...."); ACS_NVIDIA_053115 ("This bug also occurred using Viktor's MacMPI...."); POLYMATH_0013084 (Dean Dauger stating: "I use Viktor's MacMPI"); ACS_NVIDIA_005607, at _5609 ("MacMPI, freely available from the AppleSeed site

⁴ "Mathpooch, PoochMPI, semath, and SEM are different names for the same parallelization framework. Similarly, Mathpooch module, PoochMPI module, semath module, and SEM module are different names for particular modules of Mathpooch, PoochMPI, semath, and SEM on a given node." NVIDIA-ACS-0228471 (Declaration of Dean Dauger), at _481.

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

(<http://exodus.physics.ucla.edu/appleseed/>), is Viktor Decyk's 45-routine subset of MPI implemented via the Mac OS's networking APIs.").

MacMPI was based on the "Program to Program Communications Tool Box, also known as the PPC Toolbox," which was a built-in Macintosh feature. DR_NVIDIA_000713, at _716. "Viktor Decyk at UCLA hit upon the idea to use the PPC Toolbox as the basis for a library that passes messages between processors and decided to use the MPI (Message-Passing Interface) library's standard programming interface used on a wide variety of existing parallel platforms." *Id.*; see also NVIDIA-ACS-0165956.

MacMPI, via the PPC Toolbox, is designed to use AppleTalk. See, e.g., DR_NVIDIA_000703 (Launch Den Mother-Puppy Readme), at _703 ("Requirements: Power Macintoshes using an AppleTalk network...running Mac OS 8 or later and MacMPI."); DR_NVIDIA_000006 (PoochManual), at _046 ("The Launch Den Mother discovered the existence and addresses of nodes using AppleTalk calls"); *Id.* ("Once LP was up, LDM communicated with it via the Program -to-Program Communications Toolbox (PPC Toolbox), introduced in 1990 to support AppleEvents and AppleScript, written to use AppleTalk only, and toggled via the aforementioned Program Linking button."); DR_NVIDIA_000067 (PoochManualX.1); DR_NVIDIA_000113 (PoochManual); DR_NVIDIA_000160 (PoochManual); DR_NVIDIA_000247 (PoochManual); DR_NVIDIA_000309 (PoochManualX.1); DR_NVIDIA_000354 (PoochManual); DR_NVIDIA_000400 (PoochManual); DR_NVIDIA_000468 (PoochManual); DR_NVIDIA_000503 (PoochManual); DR_NVIDIA_000537 (PoochManualX); DR_NVIDIA_000571 (PoochManual); DR_NVIDIA_000638 (PoochManual); DR_NVIDIA_000703 (Launch Den Mother_Puppy README); DR_NVIDIA_000723 (PoochManual); DR_NVIDIA_000834 (PoochManual);

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

DR_NVIDIA_000948 (PoochManual); DR_NVIDIA_001048 (PoochManual);
 DR_NVIDIA_001150 (PoochManual); DR_NVIDIA_001262 (PoochManual);
 DR_NVIDIA_001746 (Dauger Research Vault - Ten Years); DR_NVIDIA_001894
 (PoochManual).

See also:

```

1  /* Partial MPI library based on Open Transport in the Macintosh OS,
2     using TCP/IP protocol.
3     No local buffering of messages is implemented, so that all messages
4     must be received in the order sent, and receives with wildcard
5     sources are not supported. the following subroutines are implemented:
6     MPI_Init, MPI_Finalize, MPI_Send, MPI_Recv, MPI_Isend, MPI_Irecv
7     MPI_Test, MPI_Wait, MPI_Sendrecv, MPI_Ssend, MPI_Issend, MPI_Waitall
8     MPI_Waitany, MPI_Get_count, MPI_Initialized, MPI_Comm_size
9     MPI_Comm_rank, MPI_Comm_dup, MPI_Comm_split, MPI_Comm_free
10    MPI_Cart_create, MPI_Cart_coords, MPI_Cart_get, MPI_Cart_shift
11    MPI_Cart_rank, MPI_Cart_sub, MPI_Dims_create
12    MPI_Bcast, MPI_Barrier, MPI_Reduce, MPI_Scan
13    MPI_Allreduce, MPI_Gather, MPI_Allgather, MPI_Scatter, MPI_Alltoall
14    MPI_Gatherv, MPI_Allgatherv, MPI_Scatterv, MPI_Alltoallv
15    MPI_Reduce_scatter, MPI_Abort, MPI_Wtime, MPI_Wtick, MPI_Type_extent
16    MPI_Request_free, MPI_Get_processor_name, MPI_Errhandler_set
17    Open Transport is described in Inside Macintosh: Networking with Open
18    Transport, version 1.3 [Apple Computer, Cupertino, CA, 1997], and at:
19    http://developer.apple.com/techpubs/mac/NetworkingOT/NetworkingWOT-2.
20    html
21    The Message Passing Interface (MPI) is described in the reference,
22    M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra,
23    MPI: The Complete Reference [MIT Press, Cambridge, MA, 1996].
24    The file MPIerrs is used throughout for error messages
25    written by viktor k. decyk, ucla
26    copyright 1999, regents of the university of california.
27    all rights reserved.
28    no warranty for proper operation of this software is given or implied.
29    software or information may be copied, distributed, and used at own
30    risk; it may not be distributed without this notice included verbatim
31    with each file.
32    update: april 1, 2004                                     */

```

POLYMATH_0003561 (MacMPI_X);

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

```

1  /* This is a Fortran callable interface library to the C version of a
2  Partial MPI library based on the Program-to-Program Communications
3  ToolBox in the Macintosh OS. No local buffering of messages is
4  implemented, so that all messages must be received in the order sent,
5  and receives with wildcard sources are not supported.
6  the following subroutines are implemented:
7  MPI_Init, MPI_Finalize, MPI_Send, MPI_Recv, MPI_Isend, MPI_Irecv
8  MPI_Test, MPI_Wait, MPI_Sendrecv, MPI_Ssend, MPI_Issend, MPI_Waitall
9  MPI_Waitany, MPI_Get_count, MPI_Initialized, MPI_Comm_size
10 MPI_Comm_rank, MPI_Comm_dup, MPI_Comm_split, MPI_Comm_free
11 MPI_Cart_create, MPI_Cart_coords, MPI_Cart_get, MPI_Cart_shift
12 MPI_Cart_rank, MPI_Cart_sub, MPI_Dims_create
13 MPI_Bcast, MPI_Barrier, MPI_Reduce, MPI_Scan
14 MPI_Allreduce, MPI_Gather, MPI_Allgather, MPI_Scatter, MPI_Alltoall
15 MPI_Gatherv, MPI_Allgatherv, MPI_Scatterv, MPI_Alltoallv
16 MPI_Reduce_scatter, MPI_Abort, MPI_Wtime, MPI_Wtick, MPI_Type_extent
17 written by viktor k. decyk, ucla
18 copyright 1998, regents of the university of california
19 update: march 28, 2003
*/

```

POLYMATH_0003563 (MacMPIf77); POLYMATH_0002091 (MacMPIf77.o);
POLYMATH_0002069 (MacMPIf77.c); POLYMATH_0013140 (MacMPI_S);
POLYMATH_0013141 (MacMPI_X); POLYMATH_0013142 (MacMPIcf);
POLYMATH_0006472 (MacMPIcf); POLYMATH_0006474 (MacMPI_X);
POLYMATH_0003532 (Dean Dager referring to MacMPI as “Viktor’s MacMPI”);
POLYMATH_0003514; POLYMATH_0003519 (Viktor Decyk discussing differenced between
MPI standard and MacMPI); POLYMATH_0003536 (Dean Dager explaining a 0-1-2-3-0
message passing patterns on MacMPI); POLYMATH_0003537 (same); POLYMATH_0002134
(trans_viktest.sit); POLYMATH_0003559 (explaining the next update to Pooch will include
certain additional MacMPI features); DR_NVIDIA_000703 (Launch Den Mother-Puppy
Readme), at _703 (“The Launch Den Mother and Launch Puppy are a pair of Macintosh
applications designed to assist in starting up parallel executables compiled with the MacMPI.f
library.”); DR_NVIDIA_000713; NVIDIA-ACS-1116010 (MacMPI_X.c); NVIDIA-ACS-
1116011 (MacMPI_X.f); NVIDIA-ACS-1116012 (MacMPIf77.c); NVIDIA-ACS-1116013
(MacMPIg77.c); NVIDIA-ACS-0237212; NVIDIA-ACS-1111707, at _707 (“MacMPI_X

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

contains a communications visualization monitor that visually describes live information about the communication pattern of your cluster.”).

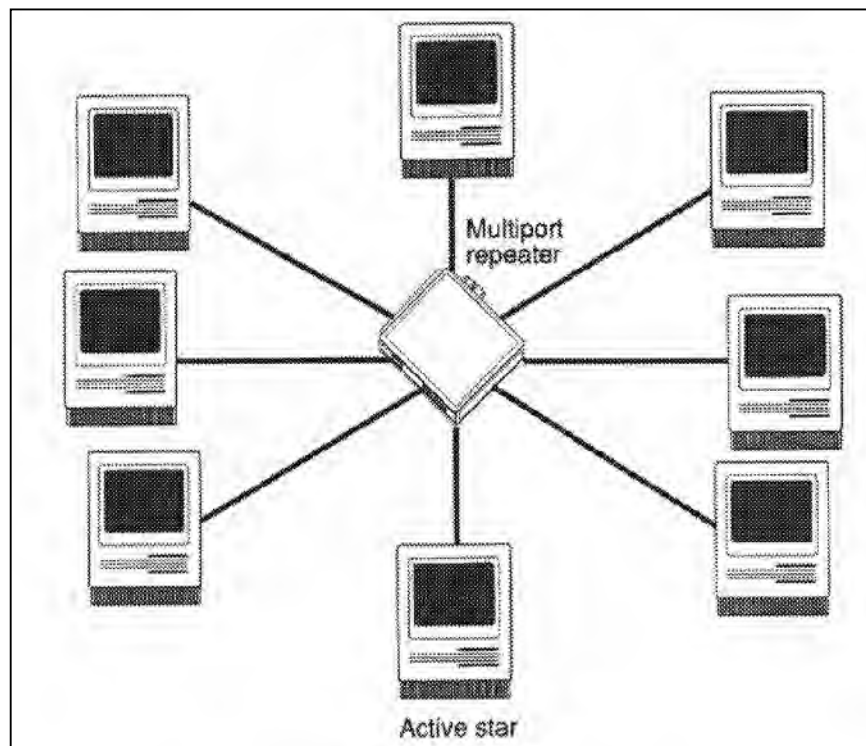
AppleTalk allowed for, *inter alia*:

Node-to-Node Communication. See NVIDIA-ACS-1320503 (Inside Macintosh – Networking), at _810 (“The protocol implementations at the physical and data-link layers of the AppleTalk protocol stack provide node-to-node delivery of data on the internet. DDP is a client of the link-access protocol-whether LLAP, ELAP, TLAP, or FDDILAP-and it uses the node-to-node delivery services provided by the data link to send and receive data. DDP is responsible for delivering data from socket to socket over an AppleTalk internet.”); NVIDIA-ACS-1332024 (Programming with AppleTalk), at _047 (“The network layer is built upon the data link layer. It is responsible for routing packets of data from node to node.”); NVIDIA-ACS-1331064 (Inside AppleTalk (2nd Ed 1990)), at _102 (“AppleTalk extends the node-to-node packet delivery service of the various individual links and the routers to a process-to-process, best-effort delivery. Thus, the various processes operating in the nodes of an internet can exchange data packets.”); “NVIDIA-ACS-1331064 (Inside AppleTalk (2nd Ed 1990)), at _119 (AppleTalk’s node-to-node packet transmission is the responsibility of the Datagram Delivery Protocol (DDP). DDP was designed to be data-link independent. This means that DDP can send its packets through any datalink and physical technology.”); NVIDIA-ACS-1331064 (Inside AppleTalk (2nd Ed 1990)), at _165 (“THE LOCALTALK LINK ACCESS PROTOCOL (LIAP) and other AppleTalk data links provide a best-effort, node-to-node delivery of packets on a single AppleTalk network. The Datagram Delivery Protocol (DDP) is designed to extend this mechanism to the socket-to-socket delivery of datagrams over an AppleTalk interact. Datagrams are packets of data carried by DDP between the sockets of an Internet. An AppleTalk internet consists of one or more AppleTalk

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

networks connected by intelligent nodes referred to as Internet routers (IRs).”); NVIDIA-ACS-1331064 (Inside AppleTalk (2nd Ed 1990)), at _632 (defining “LocalTalk Link Access Protocol (LIAP)” as “the link-level protocol that manages node-to-node delivery of data in a LocalTalk environment. LLAP manages bus access, provides a node-addressing mechanism, and controls data transmission and reception, ensuring packet length and integrity.”); NVIDIA-ACS-1320503 (Inside Macintosh - Networking), at _0531-32 (“A connection-oriented network is one in which two nodes on the network, such as computers, that want to communicate must go through a connection-establishment process, which is called a handshake. This involves the exchange of predetermined signals between the nodes in which each end identifies itself to the other. Once a connection is established, the communicating applications or processes on the nodes at either end can send and receive streams of data.”).

See also:



DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

NVIDIA-ACS-1374816 (Hands-On AppleTalk), at _879 (“The multipart repeater takes signals from any of the network segments and repeats them at full strength along all the other segments, ensuring good data communications.”); *id.* at _956 (“This solution works because you can create a data path between all the nodes so the signals reach all nodes.”); NVIDIA-ACS-1331064 (Inside AppleTalk (2nd Ed 1990)), at _633 (“node: a data-link addressable entity on a network.”); NVIDIA-ACS-1374816 (Hands-On AppleTalk), at _861 (“Node is commonly referred to in two senses. Generally, a node refers to anything attached to the network. Computers, printers, bridges, gateways, repeaters, multiport repeaters, and other peripherals all fall into this category. A stricter definition of a node is a device that is assigned an address on the network so that data may be sent directly to it.”).

Peer-to-Peer Architecture. *See*:

Peer-to-peer architecture

The network system's architecture should avoid centralized control. Such control would not only increase the initial entry cost of the network system but also create a single point of failure. Furthermore, centralized control can adversely impact efficiency and in several ways reduce the user's personal control over network resources.

AppleTalk protocols are peer-to-peer in structure, and the communicating entities operate as equals when interacting.

NVIDIA-ACS-1331064 (Inside AppleTalk 2nd Ed.), at _1095; NVIDIA-ACS-1320503 (Inside Macintosh – Networking), at _521 (“AppleTalk is a network system including hardware and software that supports communication over a variety of data-link types. Using AppleTalk, applications and processes can transfer and exchange data and share resources. The central part of the AppleTalk software consists of a number of protocols arranged in layers, with each protocol offering different services.”); NVIDIA-ACS-1320503 (Inside Macintosh - Networking), at _0533 (“This session is also referred to as a peer-to-peer session. It is one in which both ends have equal

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

control over the communication. Both ends can send and receive data at the same time and initiate or terminate the session. This type of session offers more capability and is more commonly used than an asymmetrical session.”); *id.* at _0551 (“In most cases, ADSP is the protocol that Apple recommends applications use for sending and receiving data. In addition to ensuring reliable delivery of data, ADSP provides a peer-to-peer connection, that is, both ends of the connection can exert equal control over the exchange of data.”); *id.* at _0859 (“If you want to write an application that supports a peer-to-peer session in which each end of the session can send and receive data at any time, you should use the AppleTalk Data Stream Protocol (ADSP)”); *id.* at 1057 (defining “peer-to-peer communication” as “A connection in which both ends have equal control over the exchange of data and either end can begin or end the session.”); NVIDIA-ACS-1331064 (Inside AppleTalk 2nd Ed.), at _1095 (“AppleTalk protocols are peer-to-peer in structure, and the communicating entities operate as equals when interacting.”); *id.* at _1139 (discussing messages being sent between peer nodes); NVIDIA-ACS-1332024 (Programming with AppleTalk), at _2131-2170 (discussing the AppleTalk Data Stream Protocol).

Plug and Play Capability. *See:*

“Plug-and-play” capability

The user should be able to plug a computing device into a network system and use it immediately without any of the complications of configuration. This “plug-and-play” capability, pioneered in AppleTalk, has now come to be a much-sought-after convenience of network systems. Several features of AppleTalk protocols make this possible (for example, the dynamic address-acquisition capability and the use of automatic name lookup to obtain access to network resources).

NVIDIA-ACS-1331064 (Inside AppleTalk 2nd Ed.), at _1095; *see also id.* at _1098 (“On a typical network, the majority of the devices, known as network nodes, will be users’ personal computers.”); *See also generally* NVIDIA-ACS-1320503 (Inside Macintosh – Networking), at _053; *id.* at _054 (“The PPCInform, PPCRead, and PPCWrite functions should always be executed

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)

asynchronously, because they require interaction from the other application in the session before they complete execution.”); NVIDIA-ACS-1323962 (Inside Macintosh Vol. VI) (discussing various PPC Toolbox and AppleTalk communications protocols and functions that can and should be run asynchronously); NVIDIA-ACS-1320503 (Inside Macintosh – Networking), at _099.

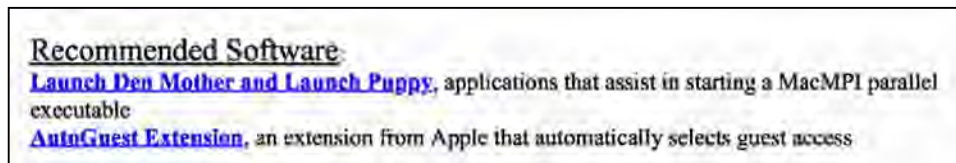
Viktor Decyk then used MacMPI to develop a computer cluster called AppleSeed. Dean Dauger allegedly assisted Viktor Decyk with AppleSeed by writing a piece of code called “Launch Den Mother” and “Launch Puppy” that helped to distribute MacMPI to each nodes on a cluster:

Originally, we had done all this manually, but that became tiresome after a while. Then I had the idea for a Launch Puppy, which would also use the PPC Toolbox, on each computer that would act as a slave node that would receive the object code and commands from the master executable to start the real slave nodes. I also wrote a piece of code in C that tested the Launch Puppy and ran it through its paces. But Decyk suggested that instead of him converting my C code to Fortran and adding it to MPI INIT, I rewrite that testing code so that it instead did the full distribution using the same nodelist file and began the master code after the slaves were up. During development of MacMPI, I revised that original testing code and added a user interface that we felt was sufficient for public distribution, and it became the Launch Den Mother.

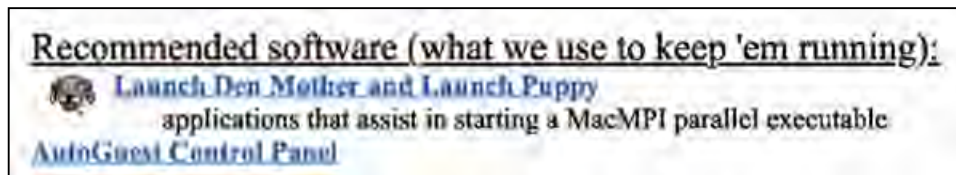
DR_NVIDIA_0007713, at _717; *see also* NVIDIA-ACS-1137686 (Apple WWDC 2004 Session 642), at 36:40-38:00; NVIDIA-ACS-1122561 (Apple Canada Video Why Macs), at 00:24-00:44 (“Viktor has a part time position over the High-Performance Computing Group at the Jet Propulsion Laboratory based in Pasadena, CA. And they have a lot of experience building Linux clusters....”).

Dean Dauger later rebranded “Launch Den Mother” and “Launch Puppy” as “Pooch.” As shown below, the UCLA webpage for AppleSeed identified “Launch Den Motion and Launch Puppy” as “applications that assist in starting a MacMPI parallel executable” from 1999 to 2001, when it started to identify “Pooch” as the “application that assists in MacMPI_IP and MacMPI_X parallel executable.”

DEFENDANTS' FINAL INVALIDITY DISCLOSURES
Advanced Cluster Systems, Inc. V. Nvidia, Case No. 1:19-Cv-2032-MN-CJB (D. Del.)



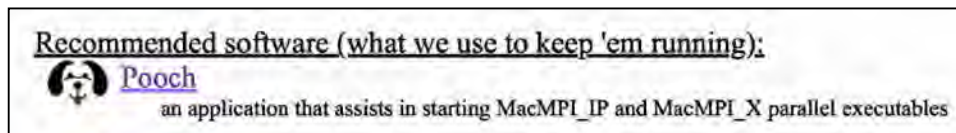
NVIDIA-ACS-1332442 (1999.04.28 - Appleseed Development Page), at _444.



NVIDIA-ACS-1332451 (2000.06.22 - Appleseed Development Page), at _451.



NVIDIA-ACS-1332514 (2001.02.01 - Appleseed Development Page)



NVIDIA-ACS-1332522 (2001.04.29 - Appleseed Development Page); *see also* NVIDIA-ACS-1332526 (2001.05.01 - AppleSeed Recipe), at _526 (“To run a program on the cluster, you should download two additional items: Pooch....”); NVIDIA-ACS-1332531; NVIDIA-ACS-0165985 (Charles Moore, *The AppleSeed Project: Clustered Power Macs Outperform Cray Supercomputer*, LOW END MAC (Apr. 19, 2000)), at _985 (“The AppleSeed team has constructed a parallel cluster.... A subset of the MPI message-passing library was implemented in Fortran77 and C. This library enabled them to port code *without modification* from other parallel processors to the Macintosh cluster. . . . Unlike Unix-based clusters, the researchers say *no special expertise in operating systems is required to build and run the cluster.*”) (emphasis in original).

EXHIBIT I

**IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE**

ADVANCED CLUSTER SYSTEMS, INC.,

Plaintiff,

v.

NVIDIA CORPORATION, NVIDIA
SIGNAPORE PTE. LTD, NVIDIA
INTERNATIONAL, INC.

Defendant.

Civil Action No. 1:19-cv-02032-MN-CJB

**DEFENDANTS' THIRD SUPPLEMENTAL RESPONSES AND OBJECTIONS TO
PLAINTIFF'S SIXTH SET OF INTERROGATORIES (NO. 10) AND SECOND
SUPPLEMENTAL RESPONSES AND OBJECTIONS
TO PLAINTIFF'S SEVENTH SET OF INTERROGATORIES (NOS. 11-20)**

Pursuant to Rules 26 and 33 of the Federal Rules of Civil Procedure, Defendant NVIDIA Corporation, NVIDIA Singapore Pte. Ltd., and NVIDIA International, Inc. ("NVIDIA" or "Defendants") provide these supplemental responses to the Sixth Set of Interrogatories (No. 10) and Seventh Set of Interrogatories (Nos. 11-20) by Plaintiff Advanced Cluster Systems, Inc. ("Plaintiff") as follows:

GENERAL STATEMENT AND OBJECTIONS

1. NVIDIA objects to each interrogatory, definition and instruction to the extent that it seeks to impose duties or obligations on NVIDIA beyond those set forth in the Federal Rules of Evidence, F.R.C.P., the Local Rules of Civil Practice and Procedure of the United States District Court for the District of Delaware ("Delaware Local Rules"), the Court's orders or any agreement reached between the parties. In this regard, to the extent that an interrogatory requires, explicitly or implicitly, the construction of one or more patent claim terms, the comparison of the asserted



NVIDIA reserves the right to supplement its response to this Interrogatory as facts and theories are adduced and developed during expert discovery and any remaining fact discovery.

ANSWERS TO THE SEVENTH SET OF INTERROGATORIES

INTERROGATORY NO. 11:

Describe the complete factual and legal bases for NVIDIA's fifth affirmative defense of prosecution history estoppel, and identify all facts, documents (including by Bates number), and witnesses concerning the foregoing.

FIRST SUPPLEMENTAL RESPONSE TO INTERROGATORY NO. 11 (SEPTEMBER 15, 2022):

NVIDIA objects to this interrogatory on the grounds set forth in its General Statement and Objections above, and hereby incorporates these by reference as if fully set forth herein. NVIDIA objects to this interrogatory on the ground that it is overly broad, unduly burdensome and compound. NVIDIA objects to this interrogatory to the extent it seeks information that is neither relevant, nor reasonably calculated to lead to the discovery of admissible evidence. NVIDIA objects to this interrogatory on the grounds that it seeks information that is outside NVIDIA's possession, custody and control or that is not maintained in the ordinary course of business. NVIDIA objects to this interrogatory as vague, ambiguous, overly broad and unduly burdensome as to the phrases "complete factual and legal bases" and "all facts, documents and witnesses." NVIDIA objects to this interrogatory to the extent that it seeks to elicit information subject to and protected by the attorney-client privilege, the attorney work product doctrine, joint defense or common interest privilege and/or any other applicable privileges, protections, or immunities. NVIDIA objects to this interrogatory on the ground that it is premature. NVIDIA objects to this interrogatory to the extent it seeks an expert opinion.



Subject to and without waiving the foregoing general and specific objections, NVIDIA responds as follows:

Plaintiff is bound by each and every statement, representation and admission made during prosecution of the '768 patent and all related patents and applications. Prosecution history estoppel occurs either (1) by making a narrowing amendment to the claim (“amendment-based estoppel”) or (2) by surrendering claim scope through argument to the patent examiner (“argument-based estoppel”).

Documents that NVIDIA may rely upon related to this defense include the prosecution history for the '768 patent as well as any *inter partes* review submissions by ACS. Indeed, as discussed further below, ACS made various amendments and statements during patent prosecution and during *inter partes* review that result in prosecution history estoppel and disclaimer of claim scope and meaning.

I. Prosecution of the '768 Patent

A. November 21, 2016 and August 8, 2017 Amendments and Remarks

While prosecuting the '768 patent, ACS made major amendments to the claims that result in prosecution history estoppel. During prosecution of the '768 patent, the Examiner several rounds of rejections. In response, ACS filed several amendments and responses/remarks, which form the basis for prosecution history estoppel.

On November 21, 2016, ACS amended prosecution claim 2 (which became '768 patent claim 1) as follows:

2. (Currently amended) A computer cluster comprising:
- a plurality of nodes comprising a hardware processor, wherein one or more of the nodes receives a command to start a cluster initialization process for the computer cluster, and wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, causes the hardware processor module configured to interpret user instructions; and
 - a mechanism for the nodes to communicate with each other using a peer-to-peer architecture;
 - wherein at least one of the nodes returns a result to a user interface or a script.

ACS_NVIDIA_000563 ('768 patent file history), at _1006 (Nov. 21, 2016, Response) (emphasis in original). Subsequently, and after the Examiner rejected the amended claims, ACS responded with additional amendments as follows to prosecution claim 2 (issued claim 1):

2. **(Currently amended)** A computer cluster comprising:
- a plurality of nodes comprising a hardware processor, wherein one or more of the nodes are configured to receive ~~receives~~ a command to start a cluster initialization process for the computer cluster, and wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, causes the hardware processor to interpret user instructions, to evaluate mathematical expressions, and to produce results of mathematical expression evaluation; and
 - a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture;
 - wherein at least one of the nodes ~~returns a result~~ is configured to return at least one result of mathematical expression evaluation to a user interface or a script.

ACS_NVIDIA_000563 ('768 patent file history), at ACS_NVIDIA_000914 (Aug. 8, 2017, Response) (emphasis in original). All of the amendments shown above that introduce new claim language are subject to prosecution history estoppel. For example, these two sets of amendments demonstrate that the claim term “results of mathematical expression evaluation with each other

using a peer-to-peer- architecture” (*see* ’487 patent at 30:33-35), which relates to what is communicated by the claimed “mechanism,” was added as a new limitation during prosecution, thereby narrowing claim scope and is subject to prosecution history estoppel. Indeed, ACS further made clear that the amendments were made to overcome prior art, stating as follows:

- (1) “For example, Block does not disclose at least a computer cluster comprising ‘a mechanism for the nodes to communicate with each other using a peer-to-peer architecture,’ in combination with the other features recited in claim 2. ... Indeed, Block does not disclose that the disclosed target nodes and backup nodes share data with one another using a peer-to-peer architecture.” ACS_NVIDIA_000563 (’768 patent file history), at ACS_NVIDIA_0001013 (Nov. 21, 2016, Remarks).
- (2) “Neither Block nor Singh discloses a computer cluster node having a mechanism for communicating results of mathematical expression evaluation with other nodes of the computer cluster” ACS_NVIDIA_000563 (’768 patent file history), at ACS_NVIDIA_000911 (Aug. 8, 2017, Remarks).

These statements by ACS further demonstrate prosecution history estoppel with respect to the claim term “results of mathematical expression evaluation with each other using a peer-to-peer-architecture.”

In addition, during the August 8, 2017 Amendments, ACS also newly added prosecution claim 37 (issued claim 35), which included claim language reciting “a mechanism to communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture.” *See* ACS_NVIDIA_000563 (’768 patent file history), at ACS_NVIDIA_000919-920 (Aug. 8, 2017, Amendments); ’768 patent at 35:23-25. For the same reasons as above, this claim term in claim 37 is also subject to prosecution history estoppel.

B. October 5, 2018 Amendments and Remarks

On October 5, 2018, ACS amended prosecution claim 2 (issued claim 1) by adding the following language:

2. (Currently amended) A computer cluster comprising:

....

wherein the plurality of nodes comprises:

a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and

a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of the first mathematical expression evaluation to a third node;

wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;

ACS_NVIDIA_000563 ('768 patent file history), at _638 (Oct. 5, 2018, Response) (emphasis in original).

At the same time, ACS made the same changes to prosecution claim 28 (which became '768 patent claim 26) and prosecution claim 31 (which became '768 patent claim 29):

28. (Currently amended) A computer cluster comprising:

....

wherein the plurality of nodes comprises:

a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and

a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of mathematical expression evaluation to a third node;

wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;

....

31. **(Currently amended)** A computer cluster comprising:

....

wherein the plurality of nodes comprises:

a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and

a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of mathematical expression evaluation to a third node;

wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node

ACS_NVIDIA_000563 ('768 patent file history), at ACS_NVIDIA_000641-44 (Oct. 5, 2018, Response) (emphasis in original).

The applicant did not originally amend prosecution claim 37 (issued claim 35) to include the language added to the other independent claims until the “[e]xaminer contacted applicant and proposed ... re-writ[ing] independent claim 37 to have all the limitations of that other independent claims ... in order to put the application in condition for allowance.” ACS_NVIDIA_000563 ('768 patent file history), at ACS_NVIDIA_000606. The applicant agreed. *Id.* As a result, the examiner amended prosecution claim 37 (issued claim 35):

37. **(Currently amended)** A computer cluster node for evaluating expressions in parallel with other computer cluster nodes, the computer cluster node comprising:

....

program code that, when executed, is capable of causing the hardware processor to: receive calls from a second node comprising a second hardware processor configured to access a second memory comprising program code for a user interface and program code for a second single-node kernel, the second single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution;

execute, using the hardware processor, at least a first mathematical expression evaluation; and

communicate a result of the first mathematical expression evaluation to a third node comprising a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the computer cluster node, execute at least a second mathematical expression evaluation using the result of the first mathematical expression evaluation, and communicate a result of the second mathematical expression evaluation to the first node;

ACS_NVIDIA_000563 ('768 patent file history), at ACS_NVIDIA_000602-03.

These amendments add specificity to the claims—they require specific nodes to perform the specific 1-2-3-1 order—and were crucial to ACS's argument that its claims should be allowed. In other words, ACS amended the claims to newly add limitations identified above to narrow claim scope and to overcome prior art rejections, thus subjecting those claim limitations to prosecution history estoppel. This is further confirmed by the statements made by ACS during prosecution, who argued to the Examiner that the cited prior art did not disclose the newly add claim limitations:

The Office Action rejects Claims 2 and 4-41 under 35 U.S.C. § 103, arguing that they are unpatentable in view of U.S. Publication No. 2005/0021751 to Block in view of U.S. Patent No. 8,601,101 to Singh and further in view of U.S. Publication No. 2003/0195938 to Howard. The Office Action rejects Claim 3 under 35 U.S.C. § 103, arguing that it is unpatentable in view of U.S. Publication No. 2005/0021751 to Block, U.S. Patent No. 8,601,101 to Singh, U.S. Publication No. 2007/0073705 to Gray, and U.S. Publication 2003/0195938 to Howard. Applicant respectfully traverses the rejections.

[REDACTED]

Applicant has amended the independent claims in an effort to advance prosecution of the application. No new matter is added. The application as filed explains that a first node of a computer cluster can be connected to a user interface or a script and configured to receive user instructions from the user interface or the script. *E.g.*, application as filed at ¶¶ [0012], [0025]. The first node can distribute calls to other cluster nodes. *E.g.*, *id.* at ¶ [0026]. The cluster nodes can evaluate expressions according to a set of rules. *E.g.*, *id.* at ¶ [0079]. The application describes an intercommunication architecture and functions that permit intermediate results to be passed from one node to another node without involvement of the first node. *E.g.*, *id.* at ¶¶ [0027], [0121]. For example, in a non-limiting embodiment, the application describes a function that allows elements near the edge of a list, which can include intermediate results of evaluation, to be copied to neighboring processors so that neighboring edges of each node partition can interact. *Id.* at ¶¶ [0080], [0137]. Results of evaluations can be communicated back to the first node and returned to the user interface or script. *E.g.*, *id.* at ¶¶ [0026], [0133].

As explained in the interview of May 15, 2018, the cited references do not disclose the combination of features recited in the claims as amended. For example, the references even if combined do not disclose a first node configured to interpret user instructions and distribute calls to other nodes for execution; a second node configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result to a third node; and a third node configured to receive the result, execute at least a second mathematical expression evaluation using the result, and communicate the result of the second mathematical expression evaluation to the first node, in combination with the other features recited in Claim 2.

Like Claim 2, independent Claims 28, 31, and 37 similarly recite subject matter not anticipated or rendered obvious by the combination of cited references. The dependent claims are patentable because each depends directly or indirectly from a patentable independent claim and because of the additional features recited in each claim. For at least these reasons, Applicant requests withdrawal of the rejections under 35 U.S.C. § 103.

ACS_NVIDIA_000563 ('768 patent file history), at ACS_NVIDIA_000649-650 (Oct. 5, 2018, Response).

C. November 29, 2018 Notice of Allowability

In the November 29, 2018 Notice of Allowability, the Examiner amended independent prosecution claims 2 (issued claim 1), 28 (issued claim 26), 31 (issued claim 29), and 37 (issued claim 35) to add the limitations from prosecution dependent claim 26 into those independent claims. The amendments are shown below:

Prosecution Claim 2 (Issued Claim 1)

wherein one or more of the nodes are configured to:

accept user instructions;

after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other;
and

after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to one or more single-node kernels.

ACS_NVIDIA_000563 ('768 patent file history), at ACS_NVIDIA_000594-595 (November 29, 2018, Examiner Amendments).

Prosecution Claim 28 (Issued Claim 26)

wherein one or more of the nodes are configured to:

accept user instructions;

after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other;
and

after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to one or more single-node kernels.

ACS_NVIDIA_000563 ('768 patent file history), at ACS_NVIDIA_000598-599 (November 29, 2018, Examiner Amendments).

Prosecution Claim 31 (Issued Claim 29)

wherein one or more of the nodes are configured to:

accept user instructions;

after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other;
and

after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to one or more single-node kernels.

ACS_NVIDIA_000563 ('768 patent file history), at ACS_NVIDIA_000600-601 (November 29, 2018, Examiner Amendments).

Prosecution Claim 37 (Issued Claim 35)

wherein the computer cluster node is configured to:

accept user instructions;

after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other;
and

after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to the single-node kernel.

ACS_NVIDIA_000563 ('768 patent file history), at ACS_NVIDIA_000603 (November 29, 2018, Examiner Amendments).



All the claim limitations identified above that were newly added via amendment are subject to prosecution history estoppel. The newly added claim limitations narrowed the claim scope was done to overcome prior art, as noted in the Examiner's Notice of Allowability:

Independent Claims 2 as amended distinguishes itself over the prior art due to the amended limitation in combination with the rest of the limitations. It is to be noted that it is the combination of all limitations that renders the claims allowable. Claims 3-25, and 27-41 are allowed based on the same reason(s).

ACS_NVIDIA_000563 ('768 patent file history), at ACS_NVIDIA_000604 (November 29, 2018, Notice of Allowability). ACS approved of these amendments as well, as stated in the summary from a November 21, 2018 interview with the Examiner: "Examiner contacted applicant and proposed incorporating dependent claim 26 into all the independent claims ... and cancel dependent claim 26 in order to put the application in condition for allowance. Applicant's representative Lance Smemoe (REG NO 66,152) has authorized the cancellation of dependent claim 26 and incorporating it's [sic] limitation [sic] to all the independent claims" ACS_NVIDIA_000563 ('768 patent file history), at ACS_NVIDIA_000607 (November 21, 2018, Examiner-Initiated Interview Summary).

II. Inter Partes Review Proceedings

In its submissions as part of the *inter partes* review of the '768 patent, Plaintiff made several statements that give rise to prosecution history estoppel. Plaintiff argued that the '768 claims "recite[] a precise order of operations involving three specific nodes": (1) the third node receives, from the second node, a result of a calculation that was performed by the second node in a different claim limitation; (2) the third node performs a different calculation based on the received result; and (3) the third node sends the new result to the first node." IPR20201-00019 POPR at 26-27. ACS argued that this "detailed limitation" required Petitioner "to show where the

prior art teaches the precise order of operations, involving three specific nodes, specified by the claim.” *Id.*

The USPTO denied institution of IPRs for the ’768 patent (IPR2021-00019 and IPR2021-00020) because of the “precise order of operation” claimed. ACS emphasized the “precise order of operations” required by the ’768 patent in almost every document it filed:

This limitation recites a ***precise order of operations*** involving three specific nodes: (1) the third node receives, from the second node, a result of a calculation that was performed by the second node in a different claim limitation; (2) the third node performs a different calculation based on the received result; and (3) the third node sends the new result to the first node. . . . The Petition fails to prove its allegation that the prior art discloses this ***detailed limitation***. To prove that allegation, the Petition would need to show where the prior art teaches ***the precise order of operations***, involving three specific nodes, specified by the claim. The Petition does not do that.

NVIDIA-ACS-0224762 (IPR2021-00019 - Patent Owner Preliminary Response), at ACS_NVIDIA_000796 (emphasis added).

This limitation recites a ***precise order of operations*** involving three specific nodes: (1) the third node receives, from the second node, a result of a calculation that was performed by the second node in a different claim limitation; (2) the third node performs a different calculation based on the received result; and (3) the third node sends the new result to the first node. . . . The Petition fails to prove its allegation that the prior art discloses this ***detailed limitation***. To prove that allegation, the Petition would need to show where the prior art teaches the ***precise order of operations***, involving three specific nodes, specified by the claim. The Petition does not do that.

NVIDIA-ACS-0224830 (IPR2021-00019 – Ex. 2001: Declaration of Jaswinder Singh), ¶¶ 66–67 (emphasis added).

Claim 26, for example, recites a precise order of operations involving three specific nodes: (1) the third node receives, from the second node, a result of a calculation that was performed by the second node in a different claim limitation; (2) the third node performs a different calculation based on the received result; and (3) the third node sends the new result to the first node. . . .

The prior art cited in the Petition does not disclose the specific order of operations between the three recited nodes of claim 26. . . . no document about Distributed Maple discloses the specific order of operations

Even if the 13 Distributed Maple references are accepted as prior art and cobbled together, they do not, individually or collectively, disclose the claimed order of operations

This limitation recites a precise order of operations involving three specific nodes: (1) the third node receives, from the second node, a result of a calculation that was performed by the second node in a different claim limitation; (2) the third node performs a different calculation based on the received result; and (3) the third node sends the new result to the first node. . . .

The Petition fails to prove its allegation that the prior art discloses this detailed limitation. To prove that allegation, the Petition would need to show where the prior art teaches the precise order of operations, involving three specific nodes, specified by the claim. The Petition does not do that.

NVIDIA-ACS-0228118 (IPR2021-00020 – Patent Owner Preliminary Response), at ACS_NVIDIA_000126, ACS_NVIDIA_000029-130, ACS_NVIDIA_000144.

Claim 26, for example, recites a precise order of operations involving three specific nodes: (1) the third node receives, from the second node, a result of a calculation that was performed by the second node in a different claim limitation; (2) the third node performs a different calculation based on the received result; and (3) the third node sends the new result to the first node.

. . .

This limitation recites a precise order of operations involving three specific nodes: (1) the third node receives, from the second node, a result of a calculation that was performed by the second node in a different claim limitation; (2) the third node performs a different calculation based on the received result; and (3) the third node sends the new result to the first node.

The Petition fails to prove its allegation that the prior art discloses this detailed limitation. To prove that allegation, the Petition would need to show where the prior art teaches the precise order of operations, involving three specific nodes, specified by the claim. The Petition does not do that.

NVIDIA-ACS-0228184 (IPR2021-00019 – Ex. 2001: Declaration of Jaswinder Singh), ¶¶ 66–67.

The Patent Trial and Appeals Board (“PTAB”) granted institution of almost every IPR filed for every patent in the family of the ’768 patent. *See* NVIDIA-ACS-0232790 (granting Institution of IPR2021-00108 for U.S. Patent No. 8,676,877); NVIDIA-ACS-0231637 (granting institution of IPR2021-00075 for U.S. Patent No. 8,140,612); NVIDIA-ACS-0223292 (granting

[REDACTED]

institution of IPR2020-01608 for U.S. Patent No. 8,082,289). The only IPRs on which the PTAB denied institution were the IPRs for the '768 patent, because of the "precise order of operations" claimed therein:

Patent Owner contends that the Petition fails to establish this limitation because it fails "to show where the prior art teaches the precise order of operations, involving three specific nodes, specified by the claim." . . . Because Petitioner has not provided sufficient objective evidence in support of this example, we determine that Petitioner has not shown that this example discloses the third node limitation.

NVIDIA-ACS-0228571 (Decision Denying Institution of IPR2021-00020), at ACS_NVIDIA_000595, ACS_NVIDIA_000579.

Patent Owner contends that the Petition fails to establish this limitation because it fails "to show where the prior art teaches the precise order of operations, involving three specific nodes, specified by the claim." . . . We agree with Patent Owner that claim 1 requires a precise order of operations involving the recited first, second, and third nodes. . . . Petitioner's sequence of events does not satisfy this limitation because Petitioner does not direct us to evidence that the third node will receive a result from the second node, perform a second mathematical evaluation and communicate the result of the second mathematical evaluation to the first node.

NVIDIA-ACS-0225177 (Decision Denying Institution of IPR2021-00019), at ACS_NVIDIA_000201-02.

As demonstrated above, ACS' positions in the *inter partes* review create prosecution history estoppel and disclaimer of claim scope and meaning.

NVIDIA reserves the right to supplement its response to this Interrogatory as facts and theories are adduced and developed during expert discovery and any remaining fact discovery. In particular, ACS has yet to sufficiently identify any doctrine of equivalents infringement theories. To the extent ACS later sufficiently identifies or asserts infringement theories under the doctrine of equivalents (e.g., in expert reports), NVIDIA reserves the right to supplement its response to the Interrogatory and/or introduce additional prosecution history estoppel arguments in view of ACS's disclosures.

[REDACTED]

NVIDIA. The operating system image is not in memory even in Accused Products that comprise a memory and a CPU to run an operating system.

ACS states that “[t]hese programs may be stored on disk-based storage and loaded into memory on the host for execution.” Accordingly, ACS recognizes that the Accused Products (even those that may comprise “disk-based storage”) do not comprise “a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel.” This claim language requires the Accused Products to have a memory with program code for a user interface and program code for a first single-node kernel.

This is not the case for any Accused Products. [REDACTED]

[REDACTED]

[REDACTED] Accordingly, these Accused Products lack the claimed “a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel.” [REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED] Accordingly, the DGX accused products do not practice the “first memory comprising program code for a user interface and program code for a first single-node kernel” limitation for this reason as well.

ACS further alleges that unidentified “third party tools may provide user interfaces.” ACS does not allege that any such tools are present in any Accused Products.

ACS also does not take account of the fact that Accused Products commonly operate in datacenter environments where user interfaces are not included on the Accused Products. For example, the user may interact via a user interface through an entirely different system remotely connected to an Accused Product. Thus, even if such a datacenter Accused Product is configured with an installed operating system, it does not have a memory comprising program code for a user interface.

“Node comprising a hardware processor with a plurality of processing cores”

Limitations

The asserted independent claims comprise the following limitations. The non-infringement reasons summarized below apply to these limitations, and any limitations in dependent or independent asserted claims that recite, reference, or depend on the same or similar limitations.

- second node comprising a second hardware processor with a plurality of processing cores (claims 1, 26, 29)
- wherein the hardware processor comprises multiple processor cores (claim 35)
- third node comprises a third hardware processor with a plurality of processing cores (claims 1, 26, 29, 35)

As already discussed, the claimed nodes in a computer cluster are conventional processors capable of single-threaded execution. ACS has alleged that the node limitations are satisfied by a GPU. This is incorrect. First, as explained herein, GPUs are not computers and therefore are not nodes in a computer cluster. Rather, GPUs in a product like DGX-2 are all part of a single computer under the control of a single operating system instance running on the DGX CPU. Even in the context of a single computer, a GPU is an accelerator for certain computations and graphics, rather than a processor. Moreover, the accused GPUs are not capable of single-threaded execution, unlike conventional processors and computer cluster nodes disclosed and claimed in the '768 patent. Rather, as discussed herein, GPUs are designed for multi-threaded execution that takes advantage

[REDACTED]

of their multi-SM parallel architecture. Each GPU comprises an array of Streaming Multiprocessors (SMs). Each SM is designed to execute hundreds of threads concurrently. Moreover, the term “plurality of processing cores” is understood by one of ordinary skill to refer to conventional processors with a plurality of processing cores like those disclosed in the ’768 patent. ’768, 7:11-21. SMs in a GPU are not processing cores as that term is used in connection with conventional CPUs discloses and claimed in the ’768 patent. Conventional CPUs have at most a few processing cores, each of which is understood to be of a conventional CPU architecture such as X86 or Power PC. Conventional CPUs are not stream processors. Conversely, SMs are not conventional CPU processors.

“Mathematical expression evaluation” and “communicate a result” Limitations

The asserted independent claims comprise the following limitations. The non-infringement reasons summarized below apply to these limitations, and any limitations in dependent or independent asserted claims that recite, reference, or depend on the same or similar limitations.

- a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of the first mathematical expression evaluation to a third node (claims 1, 26, 29, with minor wording differences)
- wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node (claims 1, 26, 29)
- program code that, when executed, is capable of causing the hardware processor to: execute, using the hardware processor, at least a first mathematical expression evaluation (claim 35)
- communicate a result of the first mathematical expression evaluation to a third node comprising a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the computer cluster node, execute at least a second mathematical expression evaluation using the

EXHIBIT J
Redacted in its Entirety

EXHIBIT K
Redacted in its Entirety

EXHIBIT L
Redacted in its Entirety

EXHIBIT M

STEPHEN WOLFRAM

THE **MATHEMATICA**[®] BOOK 5 7TH EDITION

*The definitive best-selling
presentation of Mathematica
by the creator of the system*

Library of Congress Cataloging-in-Publication Data

Wolfram, Stephen, 1959 –

Mathematica book / Stephen Wolfram. — 5th ed.

p. cm.

Includes index.

ISBN 1–57955–022–3 (hardbound).

1. Mathematica (Computer file) 2. Mathematics—Data processing.

I. Title.

QA76.95.W65 2003

510 .285 5369—dc21XX—XXXXX

CIP

Comments on this book will be welcomed at:

comments@wolfram.com

In publications that refer to the *Mathematica*
system, please cite this book as:

Stephen Wolfram, *The Mathematica Book*, 5th ed.

(Wolfram Media, 2003)

*First and second editions published by Addison-Wesley Publishing Company
under the title Mathematica: A System for Doing Mathematics by Computer.*

Third and fourth editions co-published by Wolfram Media and Cambridge University Press.

Published by:

WOLFRAM
MEDIA

ISBN 1–57955–022–3

Wolfram Media, Inc. web: www.wolfram-media.com; email: info@wolfram-media.com phone:
+1–217–398–9090; fax: +1–217–398–9095

mail: 100 Trade Center Drive, Champaign, IL 61820, USA

Copyright © 1988, 1991, 1996, 1999, 2003 by Wolfram Research, Inc.

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of the copyright holder.

Wolfram Research is the holder of the copyright to the *Mathematica* software system described in this book, including without limitation such aspects of the system as its code, structure, sequence, organization, “look and feel”, programming language and compilation of command names. Use of the system unless pursuant to the terms of a license granted by Wolfram Research or as otherwise authorized by law is an infringement of the copyright.

The author, Wolfram Research, Inc. and Wolfram Media, Inc. make no representations, express or implied, with respect to this documentation or the software it describes, including without limitations, any implied warranties of merchantability or fitness for a particular purpose, all of which are expressly disclaimed. Users should be aware that included in the terms and conditions under which Wolfram Research is willing to license *Mathematica* is a provision that the author, Wolfram Research, Wolfram Media, and their distribution licensees, distributors and dealers shall in no event be liable for any indirect, incidental or consequential damages, and that liability for direct damages shall be limited to the amount of the purchase price paid for *Mathematica*.

In addition to the foregoing, users should recognize that all complex software systems and their documentation contain errors and omissions. The author, Wolfram Research and Wolfram Media shall not be responsible under any circumstances for providing information on or corrections to errors and omissions discovered at any time in this book or the software it describes, whether or not they are aware of the errors or omissions. The author, Wolfram Research and Wolfram Media do not recommend the use of the software described in this book for applications in which errors or omissions could threaten life, injury or significant loss.

Mathematica, *MathLink* and *MathSource* are registered trademarks of Wolfram Research. *J/Link*, *MathLM*, *MathReader*, *.NET/Link*, *Notebooks* and *webMathematica* are trademarks of Wolfram Research. All other trademarks used are the property of their respective owners. *Mathematica* is not associated with Mathematica Policy Research, Inc. or MathTech, Inc.

Printed in the United States of America. (∞) Acid-free paper.

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

Part 1

This Part gives a self-contained introduction to *Mathematica*, concentrating on using *Mathematica* as an interactive problem-solving system.

When you have read this Part, you should have sufficient knowledge of *Mathematica* to tackle many kinds of practical problems.

You should realize, however, that what is discussed in this Part is in many respects just the surface of *Mathematica*. Underlying all the various features and capabilities that are discussed, there are powerful and general principles. These principles are discussed in Part 2. To get the most out of *Mathematica*, you will need to understand them.

This Part does not assume that you have used a computer before. In addition, most of the material in it requires no knowledge of mathematics beyond high-school level. The more advanced mathematical aspects of *Mathematica* are discussed in Part 3 of this book.

1.0 Running *Mathematica*

To find out how to install and run *Mathematica* you should read the documentation that came with your copy of *Mathematica*. The details differ from one computer system to another, and are affected by various kinds of customization that can be done on *Mathematica*. Nevertheless, this section outlines two common cases.

Note that although the details of running *Mathematica* differ from one computer system to another, the structure of *Mathematica* calculations is the same in all cases. You enter input, then *Mathematica* processes it, and returns a result.

1.0.1 Notebook Interfaces

use an icon or the Start menu	graphical ways to start <i>Mathematica</i>
<code>mathematica</code>	the shell command to start <i>Mathematica</i>
text ending with Shift-Enter	input for <i>Mathematica</i> (Shift-Return on some keyboards)
choose the Quit menu item	exiting <i>Mathematica</i>

Running *Mathematica* with a notebook interface.

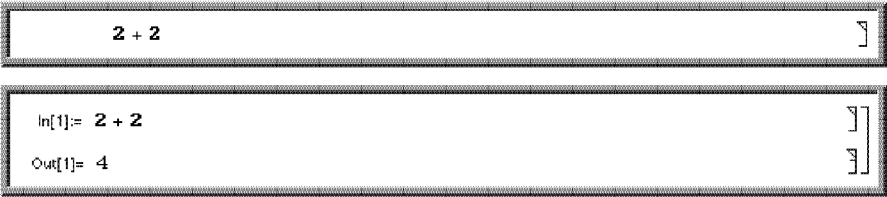
In a “notebook” interface, you interact with *Mathematica* by creating interactive documents.

If you use your computer via a purely graphical interface, you will typically double-click the *Mathematica* icon to start *Mathematica*. If you use your computer via a textually based operating system, you will typically type the command `mathematica` to start *Mathematica*.

When *Mathematica* starts up, it usually gives you a blank notebook. You enter *Mathematica* input into the notebook, then type Shift-Enter to make *Mathematica* process your input. (To type Shift-Enter, hold down the Shift key, then press Enter.) You can use the standard editing features of your graphical interface to prepare your input, which may go on for several lines. Shift-Enter tells *Mathematica* that you have finished your input. If your keyboard has a numeric keypad, you can use its Enter key instead of Shift-Enter.

After you send *Mathematica* input from your notebook, *Mathematica* will label your input with `In [n] :=`. It labels the corresponding output `Out [n] =`.

You type $2 + 2$, then end your input with Shift-Enter. *Mathematica* processes the input, then adds the input label **In[1] :=**, and gives the output.



Throughout this book, “dialogs” with *Mathematica* are shown in the following way:

With a notebook interface, you just type in $2 + 2$. *Mathematica* then adds the label **In[1] :=**, and prints the result.

In[1] := $2 + 2$

Out[1] = 4

Section 0.5.1 discusses some important details about reproducing the dialogs on your computer system. Section 1.3 gives more information about *Mathematica* notebooks.

You should realize that notebooks are part of the “front end” to *Mathematica*. The *Mathematica* kernel which actually performs computations may be run either on the same computer as the front end, or on another computer connected via some kind of network or line. In most cases, the kernel is not even started until you actually do a calculation with *Mathematica*.

To exit *Mathematica*, you typically choose the **Quit** menu item in the notebook interface.

1.0.2 Text-Based Interfaces

<code>math</code>	the operating system command to start <i>Mathematica</i>
text ending with Enter	input for <i>Mathematica</i>
Control-D or <code>Quit[]</code>	exiting <i>Mathematica</i>

Running *Mathematica* with a text-based interface.

With a text-based interface, you interact with your computer primarily by typing text on the keyboard.

To start *Mathematica* with a text-based interface, you typically type the command `math` at an operating system prompt. On some systems, you may also be able to start *Mathematica* with a text-based interface by double-clicking on a *Mathematica* Kernel icon.

When *Mathematica* has started, it will print the prompt **In[1] :=**, signifying that it is ready for your input. You can then type your input, ending with Enter or Return.

Mathematica will then process the input, and generate a result. If it prints the result out, it will label it with **Out[1] =**.

Throughout this book, dialogs with *Mathematica* are shown in the following way:

The computer prints **In[1] :=**. You just type in $2 + 2$. The line that starts with **Out[1] =** is the result from *Mathematica*.

In[1] := $2 + 2$

Out[1] = 4

Section 0.5.1 discusses some important details about reproducing the dialogs on your computer system. Note that you do not explicitly type the `In [n] :=` prompt; only type the text that follows this prompt.

Note also that most of the actual dialogs given in the book show output in the form you get with a notebook interface to *Mathematica*; output with a text-based interface looks similar, but lacks such features as special characters and font size changes.

Section 1.3 gives more details on running *Mathematica* with a text-based interface. To exit *Mathematica*, either type `Control-D`, `Control-Z` or `Quit[]` at an input prompt.

1.3 Using the Mathematica System

1.3.1 The Structure of Mathematica

Mathematica kernel	the part that actually performs computations
Mathematica front end	the part that handles interaction with the user

The basic parts of the Mathematica system.

Mathematica is a modular software system in which the *kernel* which actually performs computations is separate from the *front end* which handles interaction with the user.

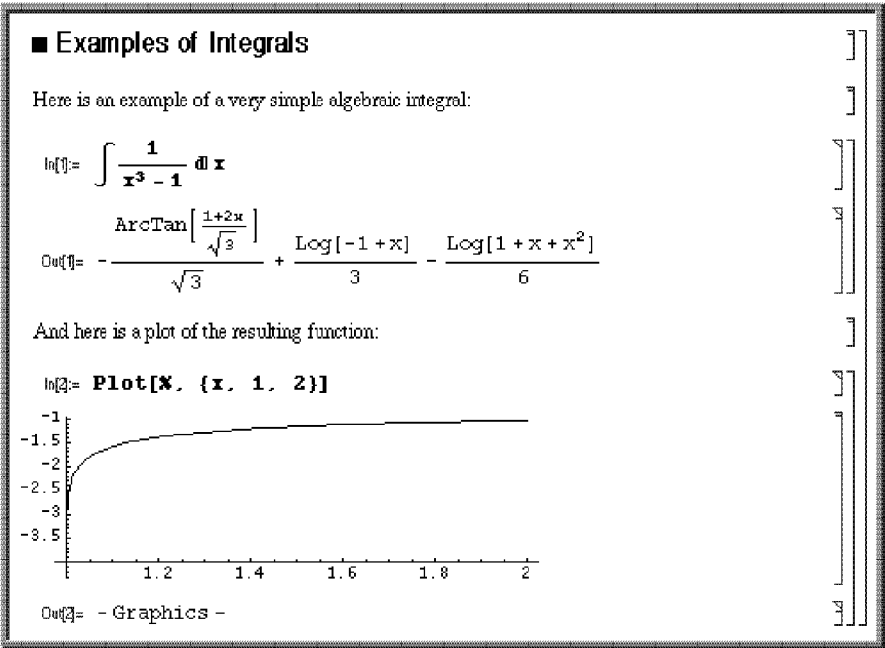
The most common type of front end for Mathematica is based on interactive documents known as *notebooks*. Notebooks mix Mathematica input and output with text, graphics, palettes and other material. You can use notebooks either for doing ongoing computations, or as means of presenting or publishing your results.

Notebook interface	interactive documents
Text-based interface	text from the keyboard
MathLink interface	communication with other programs

Common kinds of interfaces to Mathematica.

The notebook front end includes many menus and graphical tools for creating and reading notebook documents and for sending and receiving material from the Mathematica kernel.

A notebook mixing text, graphics and Mathematica input and output.



In some cases, you may not need to use the notebook front end, and you may want instead to interact more directly with the Mathematica kernel. You can do this by using a text-based interface, in which text you type on the keyboard goes straight to the kernel.

A dialog with *Mathematica* using a text-based interface.

```
In[1]:= 2^100
```

```
Out[1]= 1267650600228229401496703205376
```

```
In[2]:= Integrate[1/(x^3 - 1), x]
```

$$\text{Out[2]} = -\left(\frac{\text{ArcTan}\left[\frac{1+2x}{\sqrt{3}}\right]}{\sqrt{3}}\right) + \frac{\text{Log}[-1+x]}{3} - \frac{\text{Log}[1+x+x^2]}{6}$$

An important aspect of *Mathematica* is that it can interact not only with human users but also with other programs. This is achieved primarily through *MathLink*, which is a standardized protocol for two-way communication between external programs and the *Mathematica* kernel.

A fragment of C code that communicates via *MathLink* with the *Mathematica* kernel.

```
MLPutFunction(stdlink, "EvaluatePacket", 1);

MLPutFunction(stdlink, "Gamma", 2);
MLPutReal(stdlink, 2);
MLPutInteger(stdlink, n);

MLEndPacket(stdlink);
MLCheckFunction(stdlink, "ReturnPacket", &n);

MLGetReal(stdlink, &result);
```

Among the many *MathLink*-compatible programs that are now available, some are set up to serve as complete front ends to *Mathematica*. Often such front ends provide their own special user interfaces, and treat the *Mathematica* kernel purely as an embedded computational engine. If you are using *Mathematica* in this way, then only some parts of the discussion in the remainder of this section will probably be relevant.

1.3.2 Differences between Computer Systems

There are many detailed differences between different kinds of computer systems. But one of the important features of *Mathematica* is that it allows you to work and create material without being concerned about such differences.

In order to fit in as well as possible with particular computer systems, the user interface for *Mathematica* on different systems is inevitably at least slightly different. But the crucial point is that beyond superficial differences, *Mathematica* is set up to work in exactly the same way on every kind of computer system.

- The language used by the *Mathematica* kernel
- The structure of *Mathematica* notebooks
- The *MathLink* communication protocol

Elements of *Mathematica* that are exactly the same on all computer systems.

The commands that you give to the *Mathematica* kernel, for example, are absolutely identical on every computer system. This means that when you write a program using these commands, you can immediately take the program and run it on any computer that supports *Mathematica*.

The structure of *Mathematica* notebooks is also the same on all computer systems. And as a result, if you create a notebook on one computer system, you can immediately take it and use it on any other system.

- The visual appearance of windows, fonts, etc.
- Mechanisms for importing and exporting material from notebooks
- Keyboard shortcuts for menu commands

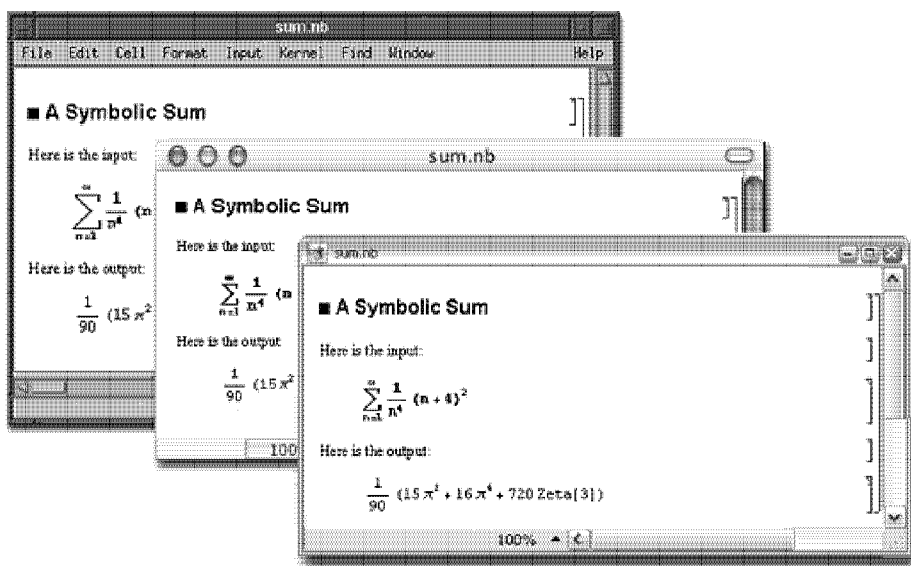
Elements that can differ from one computer system to another.

Although the underlying structure of *Mathematica* notebooks is always the same, there are often superficial differences in the way notebooks look on different computer systems, and in some of the mechanisms provided for interacting with them.

The goal in each case is to make notebooks work in a way that is as familiar as possible to people who are used to a particular type of computer system.

And in addition, by adapting the details of notebooks to each specific computer system, it becomes easier to exchange material between notebooks and other programs running on that computer system.

The same *Mathematica* notebook on three different computer systems. The underlying structure is exactly the same, but some details of the presentation are different.



One consequence of the modular nature of the *Mathematica* system is that its parts can be run on different computers. Thus, for example, it is not uncommon to run the front end for *Mathematica* on one computer, while running the kernel on a quite separate computer.

Communications between the kernel and the front end are handled by *MathLink*, using whatever networking mechanisms are available.

EXHIBIT N

Filed: February 10, 2021

Filed on behalf of

Patent Owner Advanced Cluster Systems, Inc.

By: Jon W. Gurka

Ted M. Cannon

Cheryl T. Burgess

KNOBBE, MARTENS, OLSON & BEAR, LLP

2040 Main Street, Fourteenth Floor

Irvine, CA 92614

Tel.: (949) 760-0404

Fax: (949) 760-9502

Email: BoxZTANNL.017LP2@knobbe.com

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

NVIDIA CORPORATION,
Petitioners,

v.

ADVANCED CLUSTER SYSTEMS, INC.
Patent Owner

Case No. IPR2021-00019
U.S. Patent 10,333,768

PATENT OWNER PRELIMINARY RESPONSE

TABLE OF CONTENTS

	Page No.
I. INTRODUCTION AND SUMMARY OF ARGUMENT.....	1
II. CLAIM CONSTRUCTION	6
A. “a mechanism for the nodes to communicate . . . with each other using a peer-to-peer architecture” (all challenged claims)	6
1. The “wherein” clause refers back to the same “mechanism for the nodes to communicate . . . with each other using a peer-to-peer architecture” introduced earlier in the claim.	7
2. “peer-to-peer architecture”	8
B. “cluster node module” (claim 4)	12
III. THE PETITION DOES NOT ESTABLISH A REASONABLE LIKELIHOOD THAT ANY CHALLENGED CLAIM IS UNPATENTABLE.....	17
A. The Petition does not establish that the Distributed Maple Code references are prior art printed publications.	17
B. The Petition does not establish claims 1, 4-10, 18-22, 24-25, 30-31, or 33-34 are unpatentable.	20
1. The Petition does not show that the prior art discloses the “peer-to-peer architecture” limitation.	20
2. The Petition does not show that the prior art discloses communication of “user instructions” using the “peer-to-peer architecture.”	24

TABLE OF CONTENTS (*cont'd*)

	Page No.
3. The Petition does not show that the prior art discloses “wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node.”	26
C. The Petition does not establish claims 4, 6-10, 30, or 31 are unpatentable.....	29
D. The Petition improperly relies on alleged public use evidence.....	31
E. Objective evidence confirms the non-obviousness of the challenged claims.	34
1. Patent Owner’s SEM TM and SET TM products met a long-felt but previously unmet need.	34
2. SEM TM and SET TM achieved surprising results.....	38
3. SEM TM and SET TM successfully addressed the need, where others had failed.	40
4. SEM TM and SET TM received industry praise.	40
5. SEM TM and SET TM were met by initial skepticism.	41
6. SEM TM and SET TM embody the challenged claims.	42
7. There is evidence that Petitioner copied the claimed invention.....	47

TABLE OF CONTENTS
(*cont'd*)

	Page No.
F. The Board should deny the Petition on the merits.....	49
IV. THE PETITION DOES NOT COMPLY WITH THE IPR STATUTE OR REGULATIONS	49
A. The Petition does not set forth an adequate claim construction.	49
B. The Petition improperly relies on an alleged public use that can be adjudicated properly only in district court.....	52
C. The Petition exceeds the word limit.....	56
V. CONCLUSION.....	57

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

understand the disclosed “root node scheduler”—which is necessarily involved in at least the distribution of tasks—is a “central server” or “master node.” Therefore, the cited prior art does not disclose the claimed “peer-to-peer architecture” or “cluster node modules” able to “communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.”

At least because no permissible prior art discloses these two fundamental claim limitations, the Petition should be denied. The Petition should also be denied for additional reasons, as explained below.

II. CLAIM CONSTRUCTION

A. “a mechanism for the nodes to communicate . . . with each other using a peer-to-peer architecture” (all challenged claims)

Claim 1 recites “*a mechanism for the nodes to communicate* results of mathematical expression evaluation *with each other using a peer-to-peer architecture*” and “wherein one or more of the nodes are configured to . . . communicate at least some of the user instructions using *the mechanism for the nodes to communicate with each other*” (emphases added). With respect to these two limitations, there are two claim construction issues relevant to the Board’s decision whether to institute an IPR: (1) determining that both limitations refer to the same “mechanism for the nodes to communicate . . . with each other using a

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

peer-to-peer architecture” and (2) determining the meaning of “peer-to-peer architecture.” Ex. 2001 ¶ 40.

1. The “wherein” clause refers back to the same “mechanism for the nodes to communicate . . . with each other using a peer-to-peer architecture” introduced earlier in the claim.

A POSITA would understand that the “wherein” clause uses the standard claim-drafting technique of using a shorthand phrase introduced by the definite article “the”—specifically, “the mechanism for the nodes to communicate with each other”—to refer to a limitation recited earlier in the claim. A POSITA would also understand that the *only* earlier limitation of claim 1 that recites a “mechanism” that provides antecedent basis for “the mechanism” of the “wherein” clause, is “a mechanism for the nodes to communicate . . . with each other using a peer-to-peer architecture.” Accordingly, the Board should determine that the “wherein” clause refers back to the same “mechanism for the nodes to communicate . . . with each other using a peer-to-peer architecture” recited earlier in the claim. This means that “results of mathematical expression evaluation” and “at least some of the user instructions” must be communicated using the claimed “mechanism for the nodes to communicate . . . with each other using a peer-to-peer architecture.”² Ex. 2001 ¶ 41.

² No constructions of “results of mathematical expression evaluation” and “at least some of the user instructions” are needed to decide whether to institute

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

2. “peer-to-peer architecture”

The claimed communication mechanism must use a “peer-to-peer architecture.” A POSITA would understand that, in the context of cluster computing, the adjective “peer-to-peer” ordinarily means that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node. A POSITA would also look to the specification to confirm this understanding.

As explained below, a POSITA would conclude that at least communicating tasks and data is a primary function of the “peer-to-peer architecture” of the ’768 patent. Therefore, “peer-to-peer architecture,” in the context of the ’768 patent, should be construed to require at least “an architecture in which each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.” Ex. 2001 ¶ 43.

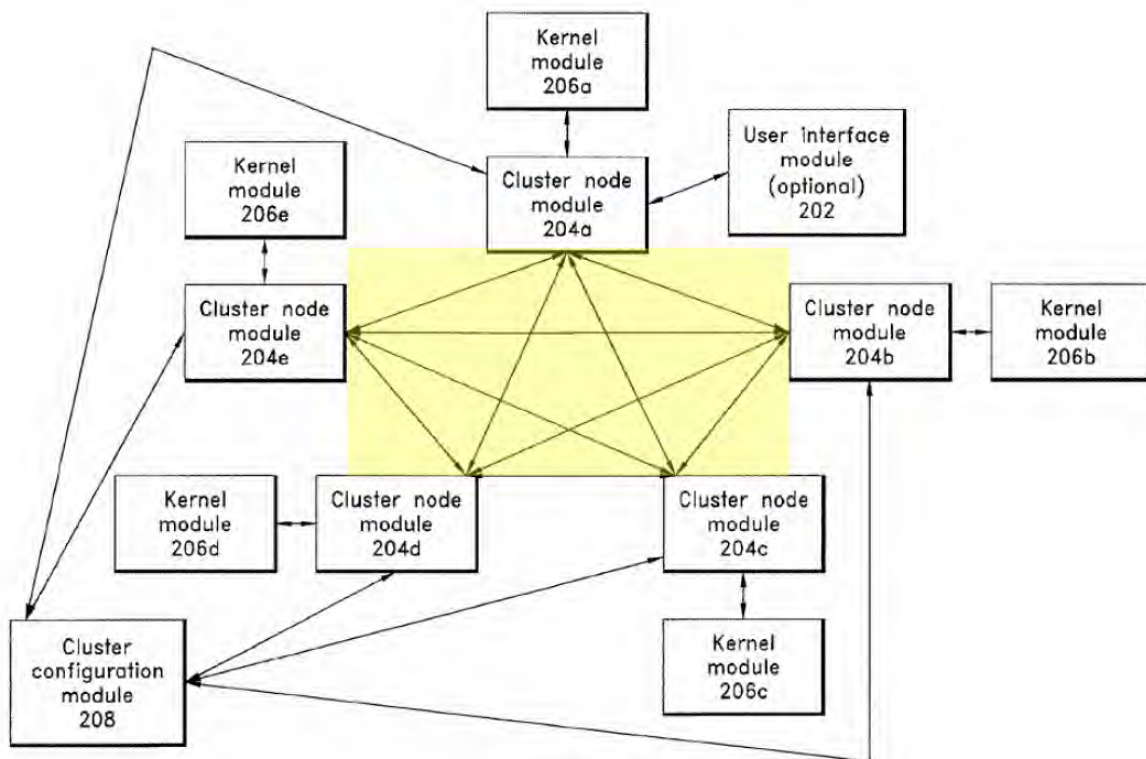
A POSITA would understand that the “peer-to-peer architecture” required by claim 1 is one feature of the interconnected “cluster node modules” disclosed by the

an IPR. However, Patent Owner reserves the right to propose constructions for these phrases in other administrative or judicial proceedings, or in the Patent Owner Response in this proceeding if an IPR is instituted.

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

'768 patent and recited by claim 4.³ Accordingly, the '768 patent's disclosure of the "peer-to-peer architecture" enabled by the "cluster node modules" is instructive of what "peer-to-peer architecture" means in the context of the '768 patent. Figure 2 illustrates that each cluster node module is connected to every other cluster node module, as shown by the yellow highlighting in the annotated figure below.

*FIG. 2*

³ Claim 1 is broader than claim 4 because, while it requires a "peer-to-peer architecture," it does not require that architecture to be implemented using the "cluster node modules" recited by claim 4.

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

Ex. 1001, Fig. 2 (highlighting added). The specification confirms that “each cluster node module 204a-e is connected to all other cluster node modules” and “[t]he cluster node modules 204a-e establish communication with one another” through “direct connections” between each cluster node module. Ex. 1001 at 23:13-14, 23:51-56. Ex. 2001 ¶ 44.

As illustrated and described, the cluster node modules provide a direct connection from each node to every other node. In addition, the specification discloses that a process for passing messages among the cluster node modules “provides the peer-to-peer behavior of the cluster node modules,” allowing them to “interact on a pair-wise or collective basis.” Ex. 1001 at 25:22-41. Ex. 2001 ¶ 45.

In view of the specification, a POSITA would understand that the message-passing process allows each cluster node module to communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node. Specifically, the specification discloses that “[c]ommunications can occur between any two or more cluster node modules” and that “[e]ach of the cluster node modules 204a-e is in communication with respective kernel modules 206a-e,” thereby enabling “MPI calls and advanced cluster commands” to be “used to parallelize program code received from an optional user interface module 208 and *distribute tasks* among the kernel modules 206a-e.” Ex. 1001 at 6:9-25 (emphasis added). Ex. 2001 ¶ 46.

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

In addition, the specification distinguishes a peer-to-peer architecture from the master-slave architecture typically used for “a form of grid computing known as ‘distributed computing.’” Ex. 1001 at 1:44-62; *see also id.* at 12:30-33 (gridMathematica connects kernels “in a master-slave relationship rather than a peer-to-peer relationship”). The specification describes the following characteristics of the master-slave architecture of grid computing:

Grid computers include at least one node known as a master node that manages a plurality of slave nodes or computational nodes. In gridMathematica, each of a plurality of kernels runs on a single node. ***One kernel is designated the master kernel, which handles all input, output, and scheduling of the other kernels (the computational kernels or slave kernels). Computational kernels receive commands and data only from the node running the master kernel.***

Id. at 1:51-59 (emphasis added). By distinguishing the peer-to-peer architecture of the invention from the master-slave architecture, the specification implicitly indicates that the peer-to-peer architecture does not have the identified characteristics of the master-slave architecture. Accordingly, by contrast to the master-slave architecture, each node in the disclosed peer-to-peer architecture is able to handle “scheduling” and communicating (including distributing, sending, and receiving) “commands and data” without requiring a central server or master node. A POSITA would understand that the communication of tasks and data falls within

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

these “scheduling” and communicating “commands and data” functions. Ex. 2001 ¶ 47.

Accordingly, a POSITA would understand that the communication of tasks and data are primary functions that the nodes of the disclosed peer-to-peer architecture of the ’768 patent can handle without requiring a central server or master node. The Board should construe “peer-to-peer architecture,” in the context of the ’768 patent, to require at least “an architecture in which each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.” Ex. 2001 ¶ 48.

B. “cluster node module” (claim 4)

Dependent claim 4 recites “wherein each of the nodes comprises one or more cluster node modules.” The phrase “cluster node module” was not a common or well-known technical phrase having any specific meaning in the relevant art at the time of the invention. Indeed, even today, more than 14 years after the earliest effective filing date of the ’768 patent, a search for patents or patent application publications that use the phrase “cluster node module” returns only patents and applications filed by the inventors of the ’768 patent. Accordingly, there was no ordinary meaning of “cluster node module” in the relevant field at the relevant time. Further, while the individual terms “cluster,” “node,” and “module,” were known in the relevant field, a POSITA would understand that no ordinary meaning for the

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

combined phrase “cluster node module” could reliably be composed by attempting to combine the meanings of the individual terms. Ex. 2001 ¶ 49.

In view of the claim language and specification, a POSITA would understand that the inventors coined the phrase “cluster node module” to encapsulate essential features of the modules that interconnect the nodes in a preferred embodiment of the invention.⁴ Because “cluster node module” is a coined phrase, reliance on the specification is necessary to ascertain its meaning. *3M Innovative Props. Co. v. Tredegar Corp.*, 725 F.3d 1315, 1321 (Fed. Cir. 2013) (“Idiosyncratic language, highly technical terms, or terms coined by the inventor are best understood by reference to the specification.”) As explained below, in view of the specification, the Board should construe “cluster node module” to mean “a module that cooperates with other cluster node modules to establish intercommunication among nodes in a computer cluster and to exchange messages such that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.” Ex. 2001 ¶ 50.

The specification discloses several *optional* components of the cluster node modules. For example:

⁴ Several claims of the ’768 patent, including claim 1, recite “a plurality of nodes” but not “cluster node modules.”

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

FIG. 3 shows one embodiment of a cluster node module 204 implementing MPI calls and advanced MPI functions. **In the embodiment shown in FIG. 3**, cluster node module 204 includes MPI module 302, advanced functions module 304, received message queue 306, and message receiving queue 308.

Ex. 1001 at 12:41-46 (emphases added). Because the components depicted in Figure 3 are part of “one embodiment,” a POSITA would understand that they are optional components of the cluster node modules. Dependent claims of U.S. Patent No. 8,082,289 (which is related to the ’768 patent) specifically reciting that the cluster node modules comprise an MPI module, advanced functions module, received message queue, and message receiving queue also demonstrate that these components are optional parts of the cluster node modules. IPR2020-01608, Ex. 1001, at claim 8 (received message queue), claim 9 (message receiving queue), claim 13 (advanced functions module), claim 26 (MPI module). Accordingly, a POSITA would understand that the basic “cluster node modules” do not require an MPI module, advanced functions module, received message queue, and message receiving queue. Thus, if the Petition suggests that a “cluster node module”

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

necessarily comprises these components (*see* Pet. at 52), the Petition is incorrect.⁵

Ex. 2001 ¶ 51.

Rather than interpreting “cluster node module” to include optional components such as those depicted by Figure 3, a POSITA would look to the specification to limit the “cluster node module” to its essential features. The specification unambiguously discloses that cluster node modules are configured to establish intercommunication with one another, communicate messages among themselves and with kernels, and, through such message-passing, “provide[] the peer-to-peer behavior of the cluster node modules.” Ex. 1001 at 23:51-52, 24:39-40, 24:52-53, 25:1-2, 25:9-10, 25:23-24, 25:37-38. Significantly, the specification does not say these disclosed features apply just to “one embodiment” of the cluster node modules or otherwise suggest they are optional. Therefore, a POSITA would understand that the capability to establish intercommunication among all cluster node modules and to exchange messages to provide “peer-to-peer behavior” are essential features of the cluster node modules. Ex. 2001 ¶ 52.

⁵ The Petition does not commit to *any* definitive claim construction of “cluster node module.” The Petition merely states that “a ‘cluster node module’ includes code *relating to* . . . and *can* also include” the components depicted by Figure 3. Pet. at 52.

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

A POSITA would further examine both the ordinary meaning of “peer-to-peer” and the specification to determine what is meant by the cluster node modules’ “peer-to-peer behavior.” As explained above with respect to the “peer-to-peer architecture” limitation, a POSITA would conclude that “peer-to-peer behavior” includes at least that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node. Ex. 2001 ¶ 53.

Accordingly, the Board should construe “cluster node module” to mean “a module that cooperates with other cluster node modules to establish intercommunication among nodes in a computer cluster and to exchange messages such that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.”⁶ Ex. 2001 ¶ 54.

⁶ A POSITA would recognize there is some overlap between the “cluster node modules” limitation of claim 4 and the “peer-to-peer architecture” limitation of claim 1, but the “cluster node modules” limitation is narrower.

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

at 316; *see also id.* at 323 (“the root is in charge of task scheduling”). Accordingly, because the communication of all tasks must go through a master node, the prior art does not disclose using a “peer-to-peer architecture” for such communication. Therefore, Petitioner failed to meet its initial burden with respect to claim 1 and its dependent claims. Ex. 2001 ¶ 64.

The Petition does not allege that it would have been obvious to modify Schreiner 1 such that the distribution of tasks does not require the root node. The Board should neither rewrite the Petition nor allow Petitioner to rewrite the Petition to include such a proposed modification. Moreover, even if the Petition had made such an obviousness allegation, it would not have been obvious to a POSITA to modify Schreiner 1 such that the distribution of tasks does not require the root node because that modification would have destroyed the ability of users to use the fault tolerance mechanism disclosed in Schreiner 1. Ex. 2001 ¶ 65.

3. **The Petition does not show that the prior art discloses “wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node.”**

This limitation recites a precise order of operations involving three specific nodes: (1) the third node receives, from the second node, a result of a calculation

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

that was performed by the second node in a different claim limitation; (2) the third node performs a different calculation based on the received result; and (3) the third node sends the new result to the first node. The Petition alleges that the alleged prior art publications “disclose this element.” Pet. at 43. Ex. 2001 ¶ 66.

Importantly, the Petition does not allege that this limitation would have been obvious; it alleges that the prior art discloses the limitation. The Board may not rewrite the Petition to change the Petition’s allegation of actual disclosure to an obviousness allegation. *SAS Inst., Inc. v. Iancu*, 138 S. Ct. 1348, 1356 (2018) (“Nothing suggests the Director enjoys a license to depart from the petition and institute a different inter partes review of his own design.”)

The Petition fails to prove its allegation that the prior art discloses this detailed limitation. To prove that allegation, the Petition would need to show where the prior art teaches the precise order of operations, involving three specific nodes, specified by the claim. The Petition does not do that. Instead, the Petition refers to the prior art’s general disclosure that multiple nodes may be involved in evaluating mathematical expressions and then speculates that the prior art could work in a manner that would meet the claim limitation. Pet. at 44-47. Ex. 2001 ¶ 67.

The Petition first relies on “the example of Figure 5” of Schreiner 1. *Id.* at 45. It is clear on the face of Petitioner’s argument that Figure 5 does not actually disclose every detail about the claim limitation and that Petitioner is merely speculating about

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

how Figure 5 allegedly could work. The Petition asserts that a “POSITA would understand that the third node then *could* execute a second mathematical expression using that result, and in fact, that is *likely why* the third node requested the result in the first place.” *Id.* at 46 (emphases added). The Petition further speculates that “the third node *could* send the result of the second mathematical expression to the first (root) node.” *Id.* (emphasis added). Ex. 2001 ¶ 68.

The Petition also relies on an “example . . . illustrated in Schreiner3 Fig. 1.” *Id.* at 47. But the cited example from Schreiner 3 is about a mechanism for “the *logging* of task return values” (Ex. 1010 at 1) and neither Schreiner 3 nor the Petition explains how Figure 1 relates to the evaluation of mathematical expressions by multiple nodes. Accordingly, Petitioner’s interpretation of Figure 1 is self-serving and speculative. Moreover, the Petition itself makes clear that Figure 1 does not actually disclose every detail about the claim limitation and that Petitioner is merely speculating about how Figure 1 allegedly could work. The Petition speculates that “once [client 1] receives the result message from node ‘client n,’ it *may* use it to perform at least a second mathematical expression evaluation.” Pet. at 47. Petitioner offers no evidence or explanation supporting this speculation. *Id.* Ex. 2001 ¶ 69.

Because the Petition merely speculates about what prior art nodes “could” or “may” do, the Petition fails to prove Petitioner’s assertion that the prior art discloses the claim limitation. Ex. 2001 ¶ 70. And because the Petition lacks any allegation

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

that it would have been obvious to modify the prior art to satisfy the claim limitation, Petitioner cannot now fall back on such an obviousness allegation. Petitioner simply failed to meet its initial burden.

C. The Petition does not establish claims 4, 6-10, 30, or 31 are unpatentable.

Claims 4, 6-10, 30, and 31 indirectly depend on claim 1 and, thus, are patentable for the same reasons set forth above that claim 1 is patentable. In addition, these claims are patentable because the prior art does not disclose the limitation of claim 4 reciting that each node “comprises one or more cluster node modules.” As explained above, because “cluster node module” is a coined phrase encapsulating the essential features of the cluster node modules disclosed in the specification, the phrase should be construed to mean “a module that cooperates with other cluster node modules to establish intercommunication among nodes in a computer cluster and to exchange messages such that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.” These essential features of the cluster node modules are fundamental parts of the claimed invention of claim 4. Ex. 2001 ¶ 71.

The Petition alleges that the “dist.Scheduler and dist.maple components” of the prior art are “cluster node modules.” Pet. at 52-55. However, the Petition does not allege or prove that those components allow each node to “communicate tasks

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

“Schreiner1,” and “¶216” and copying text as images on pages 22, 45, and 48 of the Petition. These instances add up to about 600 excess words. The Board has rejected briefs because they used similar formatting tricks to exceed page or word limits. *See, e.g., Starbucks Corp. v. Ameranth, Inc.*, CBM2015-00091, Paper 16 at 2-3 (P.T.A.B. Jan. 29, 2016) (rejecting a Patent Owner Response that used “at least 28 point” spacing rather than “double spacing”).

V. CONCLUSION

For the foregoing reasons, the Board should deny the Petition.

Respectfully submitted,

KNOBBE, MARTENS, OLSON & BEAR, LLP

Dated: February 10, 2021

By: /Ted M. Cannon/

Jon W. Gurka, Reg. No. 44,139

Ted M. Cannon, Reg. No. 55,036

Cheryl T. Burgess, Reg. No. 55,030

Attorneys for Patent Owner

Advanced Cluster Systems, Inc.

EXHIBIT O

Filed: February 10, 2021

Filed on behalf of

Patent Owner Advanced Cluster Systems, Inc.

By: Jon W. Gurka

Ted M. Cannon

Cheryl T. Burgess

KNOBBE, MARTENS, OLSON & BEAR, LLP

2040 Main Street, Fourteenth Floor

Irvine, CA 92614

Tel.: (949) 760-0404

Fax: (949) 760-9502

Email: BoxZTANNL.017LP3@knobbe.com

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

NVIDIA CORPORATION,
Petitioners,

v.

ADVANCED CLUSTER SYSTEMS, INC.
Patent Owner

Case No. IPR2021-00020
U.S. Patent 10,333,768

PATENT OWNER PRELIMINARY RESPONSE

TABLE OF CONTENTS

	Page No.
I. INTRODUCTION AND SUMMARY OF ARGUMENT.....	1
II. CLAIM CONSTRUCTION	6
A. “a mechanism to communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture” (claim 35).....	6
1. The “wherein” clause refers back to the same “mechanism to communicate . . . using a peer-to-peer architecture” introduced earlier in the claim.....	7
2. “peer-to-peer architecture”.....	8
B. “cluster node module” (claim 27)	12
III. THE PETITION DOES NOT ESTABLISH A REASONABLE LIKELIHOOD THAT ANY CHALLENGED CLAIM IS UNPATENTABLE.....	15
A. The Petition does not establish that the Distributed Maple Code references are prior art printed publications.	15
B. The Petition does not establish claim 26 is unpatentable.....	18
C. The Petition does not establish claim 27 is unpatentable.....	21
D. The Petition does not establish claim 29 is unpatentable.....	23
E. The Petition does not establish claim 35 is unpatentable.....	23
1. The Petition does not show that the prior art discloses the “peer-to-peer architecture” limitation.	23
2. The Petition does not establish the prior art discloses communication of “user instructions” using the “peer-to-peer architecture.”	28

TABLE OF CONTENTS (*cont'd*)

	Page No.
F. The Petition improperly relies on alleged public use evidence.....	30
G. Objective evidence confirms the non-obviousness of the challenged claims.	33
1. Patent Owner’s SEM TM and SET TM products met a long-felt but previously unmet need.	33
2. SEM TM and SET TM achieved surprising results.	37
3. SEM TM and SET TM successfully addressed the need, where others had failed.	39
4. SEM TM and SET TM received industry praise.	39
5. SEM TM and SET TM were met by initial skepticism.	40
6. SEM TM and SET TM embody the challenged claims.	41
7. There is evidence that Petitioner copied the claimed invention.....	46
H. The Board should deny the Petition on the merits.	48
IV. THE PETITION DOES NOT COMPLY WITH THE IPR STATUTE OR REGULATIONS	48
A. The Petition does not set forth an adequate claim construction.	48
B. The Petition improperly relies on an alleged public use that can be adjudicated properly only in district court.....	51
C. The Petition exceeds the word limit.....	55
V. CONCLUSION.....	56

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

master node. Specifically, the prior art discloses: “All remote schedulers send new tasks to the root node scheduler which distributes them among all machines.” Ex. 1108 at 316. A POSITA would understand the disclosed “root node scheduler”—which is necessarily involved in at least the distribution of tasks—is a “central server” or “master node.” Therefore, the cited prior art does not disclose the claimed “peer-to-peer architecture” or “cluster node modules” able to “communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.”

At least because no permissible prior art discloses the fundamental claim limitations, the Petition should be denied. The Petition should also be denied for additional reasons, as explained below.

II. CLAIM CONSTRUCTION

A. “a mechanism to communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture” (claim 35)

Claim 35 recites “*a mechanism to communicate* results of evaluation with other computer cluster nodes *using a peer-to-peer architecture*” and “wherein the computer cluster node is configured to . . . communicate at least some of the user instructions using *the mechanism for the nodes to communicate with each other*” (emphases added). With respect to these two limitations, there are two claim construction issues relevant to the Board’s decision whether to institute an IPR: (1)

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

determining that both limitations refer to the same “mechanism to communicate . . . using a peer-to-peer architecture” and (2) determining the meaning of “peer-to-peer architecture.” Ex. 2001 ¶ 40.

1. The “wherein” clause refers back to the same “mechanism to communicate . . . using a peer-to-peer architecture” introduced earlier in the claim.

A POSITA would understand that the “wherein” clause uses the standard claim-drafting technique of using a shorthand phrase introduced by the definite article “the”—specifically, “the mechanism for the nodes to communicate with each other”—to refer to a limitation recited earlier in the claim. A POSITA would also understand that the *only* earlier limitation of claim 1 that recites a “mechanism” that provides antecedent basis for “the mechanism” of the “wherein” clause, is “a mechanism to communicate . . . using a peer-to-peer architecture.” Accordingly, the Board should determine that the “wherein” clause refers back to the same “mechanism to communicate . . . using a peer-to-peer architecture” recited earlier in the claim. This means that “results of evaluation” and “at least some of the user instructions” must be communicated using the claimed “mechanism to communicate . . . using a peer-to-peer architecture.”² Ex. 2001 ¶ 41.

² No constructions of “results of evaluation” and “at least some of the user instructions” are needed to decide whether to institute an IPR. However, Patent

IPR2021-00020

*NVIDIA Corp. v. Advanced Cluster Sys.***2. “peer-to-peer architecture”**

The claimed communication mechanism must use a “peer-to-peer architecture.” A POSITA would understand that, in the context of cluster computing, the adjective “peer-to-peer” ordinarily means that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node. A POSITA would also turn to the specification to confirm this understanding.

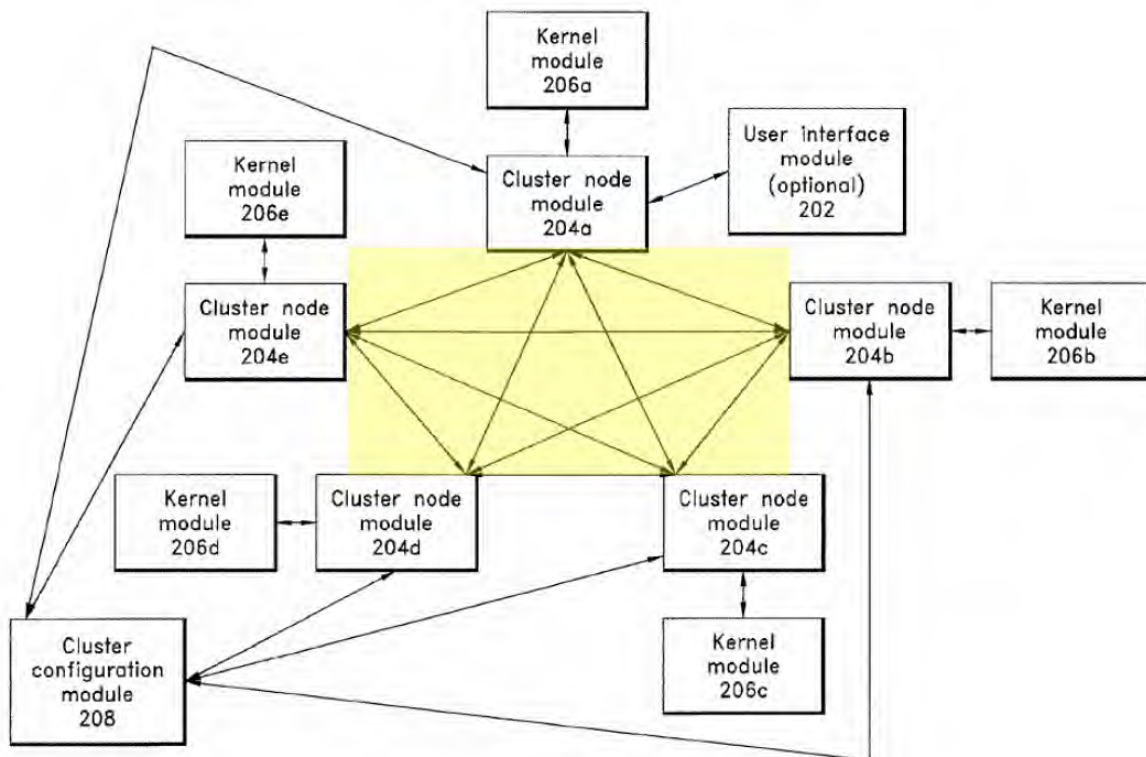
As explained below, a POSITA would conclude from the specification that communicating tasks and data with other nodes without the tasks and data being required to go through a central server or master node is a primary function of the “peer-to-peer architecture” of the ’768 patent. Therefore, “peer-to-peer architecture,” in the context of the ’768 patent, should be construed to require at least “an architecture in which each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.” Ex. 2001 ¶ 43.

Owner reserves the right to propose constructions for these phrases in other administrative or judicial proceedings, or in the Patent Owner Response in this proceeding if an IPR is instituted.

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

A POSITA would understand that the “peer-to-peer architecture” required by claim 35 is one feature of the interconnected “cluster node modules” disclosed by the ’768 patent and recited by claim 27. Accordingly, the ’768 patent’s disclosure of the “peer-to-peer architecture” enabled by the “cluster node modules” is instructive of what “peer-to-peer architecture” means in the context of the ’768 patent. Figure 2 illustrates that each cluster node module is connected to every other cluster node module, as shown by the yellow highlighting in the annotated figure below.

*FIG. 2*

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

Ex. 1101, Fig. 2 (highlighting added). The specification confirms that “each cluster node module 204a-e is connected to all other cluster node modules” and “[t]he cluster node modules 204a-e establish communication with one another” through “direct connections” between each cluster node module. Ex. 1101 at 23:13-14, 23:51-56. Ex. 2001 ¶ 44.

As illustrated and described, the cluster node modules provide a direct connection from each node to every other node. In addition, the specification discloses that a process for passing messages among the cluster node modules “provides the peer-to-peer behavior of the cluster node modules.” Ex. 1101 at 25:22-38. Ex. 2001 ¶ 45.

In view of the specification, a POSITA would understand that the message-passing process allows each cluster node module to communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node. Specifically, the specification discloses that “[c]ommunications can occur between any two or more cluster node modules” and that “[e]ach of the cluster node modules 204a-e is in communication with respective kernel modules 206a-e,” thereby enabling “MPI calls and advanced cluster commands” to be “used to parallelize program code received from an optional user interface module 208 and *distribute tasks* among the kernel modules 206a-e.” Ex. 1101 at 6:9-25 (emphasis added). Ex. 2001 ¶ 46.

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

In addition, the specification distinguishes a peer-to-peer architecture from the master-slave architecture typically used for “a form of grid computing known as ‘distributed computing.’” Ex. 1101 at 1:44-62; *see also id.* at 12:30-33 (gridMathematica connects kernels “in a master-slave relationship rather than a peer-to-peer relationship”). The specification describes the following characteristics of the master-slave architecture of grid computing:

Grid computers include at least one node known as a master node that manages a plurality of slave nodes or computational nodes. In gridMathematica, each of a plurality of kernels runs on a single node. ***One kernel is designated the master kernel, which handles all input, output, and scheduling of the other kernels (the computational kernels or slave kernels). Computational kernels receive commands and data only from the node running the master kernel.***

Id. at 1:51-59 (emphasis added). By distinguishing the peer-to-peer architecture of the invention from the master-slave architecture, the specification implicitly indicates that the peer-to-peer architecture does not have the identified characteristics of the master-slave architecture. Accordingly, by contrast to the master-slave architecture, each node in the disclosed peer-to-peer architecture is able to handle “scheduling” and communicating (including distributing, sending, and receiving) “commands and data” without requiring a central server or master node. A POSITA would understand that the communication of tasks and data falls within

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

these “scheduling” and communicating “commands and data” functions. Ex. 2001 ¶ 47.

Accordingly, a POSITA would understand that the communication of tasks and data are primary functions that the nodes of the disclosed peer-to-peer architecture of the ’768 patent can handle without requiring a central server or master node. The Board should construe “peer-to-peer architecture,” in the context of the ’768 patent, to require at least “an architecture in which each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.” Ex. 2001 ¶ 48.

B. “cluster node module” (claim 27)

Claim 27 recites “wherein the single-node kernel is configured to send the first packet to a local cluster node module.” The phrase “cluster node module” was not a common or well-known technical phrase having any specific meaning in the relevant art at the time of the invention. Indeed, even today, more than 14 years after the earliest effective filing date of the ’768 patent, a search for patents or patent application publications that use the phrase “cluster node module” returns only patents and applications filed by the inventors of the ’768 patent. Accordingly, there was no ordinary meaning of “cluster node module” in the relevant field at the relevant time. Further, while the individual terms “cluster,” “node,” and “module,” were known in the relevant field, a POSITA would understand that no ordinary

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

meaning for the combined phrase “cluster node module” could reliably be composed by attempting to combine the meanings of the individual terms. Ex. 2001 ¶ 49.

In view of the claim language and specification, a POSITA would understand that the inventors coined the phrase “cluster node module” to encapsulate essential features of the modules that interconnect the nodes in a preferred embodiment of the invention.³ Because “cluster node module” is a coined phrase, reliance on the specification is necessary to ascertain its meaning. *3M Innovative Props. Co. v. Tredegar Corp.*, 725 F.3d 1315, 1321 (Fed. Cir. 2013) (“Idiosyncratic language, highly technical terms, or terms coined by the inventor are best understood by reference to the specification.”) As explained below, in view of the specification, the Board should construe “cluster node module” to mean “a module that cooperates with other cluster node modules to establish intercommunication among nodes in a computer cluster and to exchange messages such that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.” Ex. 2001 ¶ 50.

A POSITA would not interpret “cluster node module” to include optional components disclosed in connection with only some embodiments, such as those

³ Several claims of the ’768 patent, including claim 1, recite “a plurality of nodes” but not “cluster node modules.”

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

depicted by Figure 3. Ex. 2001 ¶ 51. Instead, a POSITA would look to the specification to limit the “cluster node module” to its essential features. The specification unambiguously discloses that cluster node modules are configured to establish intercommunication with one another, communicate messages among themselves and with kernels, and, through such message-passing, “provide[] the peer-to-peer behavior of the cluster node modules.” Ex. 1101 at 25:37-38; *id.* at 23:51-52, 24:39-40, 24:52-53, 25:1-2, 25:9-10, 25:23-24. Significantly, the specification does not say these disclosed features apply just to “one embodiment” of the cluster node modules or otherwise suggest they are optional. Therefore, a POSITA would understand that the capability to establish intercommunication among all cluster node modules and to exchange messages to provide “peer-to-peer behavior” are essential features of the cluster node modules. Ex. 2001 ¶ 52.

A POSITA would further examine both the ordinary meaning of “peer-to-peer” and the specification to determine what is meant by the cluster node modules’ “peer-to-peer behavior.” As explained above with respect to the “peer-to-peer architecture” limitation, a POSITA would conclude that “peer-to-peer behavior” includes at least that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node. Ex. 2001 ¶ 53.

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

Accordingly, the Board should construe “cluster node module” to mean “a module that cooperates with other cluster node modules to establish intercommunication among nodes in a computer cluster and to exchange messages such that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.” Ex. 2001 ¶ 54.

III. THE PETITION DOES NOT ESTABLISH A REASONABLE LIKELIHOOD THAT ANY CHALLENGED CLAIM IS UNPATENTABLE

A. The Petition does not establish that the Distributed Maple Code references are prior art printed publications.

All six grounds of the Petition rely on a collection of *seven* references that the Petition calls “Distributed Maple Code.” Pet. at 17 (for Ground 1, “[a]ll elements of claims . . . are disclosed in and rendered obvious by Schreiner 1 in view of Schreiner 2, Schreiner 3, the *Distributed Maple Code*, the Maple Guide, the SPARC IV Article, and the AMD Article”) (emphasis added), 65, 68, 74, 75, 77 (each of Grounds 2-6 directly or indirectly incorporating the references in Ground 1), 9 (explaining that Distributed Maple Code includes Exhibits 1112-1118). Thus, Ground 1 is a *13*-reference obviousness ground and each of Grounds 2-6 is an obviousness ground combining between 14 and 17 references.

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

While Schreiner says his software was “accessible through Google searches” (Ex. 1106 ¶ 21), he has not established that the software would have been listed reasonably prominently in response to a search for the terms that a reasonably diligent searcher interested and ordinarily skilled in the relevant field would have employed. The most the evidence submitted by Petitioner shows is that Schreiner himself, or possibly others who helped create the Distributed Maple Code or already knew of its existence, may have been able to locate whatever version was posted at that time. Because that is insufficient to meet Petitioner’s burden to show public accessibility of the Distributed Maple Code, Petitioner cannot rely on those references to meet its burden. And because all grounds rely on the Distributed Maple Code, the Board may deny the Petition on that basis alone.

B. The Petition does not establish claim 26 is unpatentable.

The Petition does not show that the prior art discloses “wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node.” Ex. 2001 ¶ 55.

This limitation recites a precise order of operations involving three specific nodes: (1) the third node receives, from the second node, a result of a calculation

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

that was performed by the second node in a different claim limitation; (2) the third node performs a different calculation based on the received result; and (3) the third node sends the new result to the first node. The Petition relies on Schreiner 1 and Schreiner 3 for this limitation. Pet. at 47-50. Ex. 2001 ¶ 56.

Importantly, the Petition does not allege that this limitation would have been obvious; it alleges that the prior art discloses the limitation. The Board may not rewrite the Petition to change the Petition's allegation of actual disclosure to an obviousness allegation. *SAS Inst., Inc. v. Iancu*, 138 S. Ct. 1348, 1356 (2018) ("Nothing suggests the Director enjoys a license to depart from the petition and institute a different inter partes review of his own design.")

The Petition fails to prove its allegation that the prior art discloses this detailed limitation. To prove that allegation, the Petition would need to show where the prior art teaches the precise order of operations, involving three specific nodes, specified by the claim. The Petition does not do that. Instead, the Petition refers to the prior art's general disclosure that multiple nodes may be involved in evaluating mathematical expressions and then speculates that the prior art could work in a manner that would meet the claim limitation. Pet. at 44-47. Ex. 2001 ¶ 57.

The Petition first relies on "the example of Figure 5" of Schreiner 1. Pet. at 48. It is clear on the face of Petitioner's argument that Figure 5 does not actually disclose every detail about the claim limitation and that Petitioner is merely

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

speculating about how Figure 5 allegedly could work. The Petition asserts that a “POSITA would understand that the third node then *could* execute a second mathematical expression using that result, and in fact, that is *likely why* the third node requested the result in the first place.” *Id.* at 48-49 (emphases added). The Petition further speculates that “the third node *could* send the result of the second mathematical expression to the first (root) node.” *Id.* at 49 (emphasis added). Ex. 2001 ¶ 58.

The Petition also relies on an “example . . . illustrated in Schreiner 3 Fig. 1.” *Id.* at 49. But the cited example from Schreiner 3 is about a mechanism for “the *logging* of task return values” (Ex. 1110 at 1) and neither Schreiner 3 nor the Petition explains how Figure 1 relates to the evaluation of mathematical expressions by multiple nodes. Accordingly, Petitioner’s interpretation of Figure 1 is self-serving and speculative. Moreover, the Petition itself makes clear that Figure 1 does not actually disclose every detail about the claim limitation and that Petitioner is merely speculating about how Figure 1 allegedly could work. The Petition speculates that “once [client 1] receives the result message from node ‘client n,’ it *may* use it to perform at least a second mathematical expression evaluation.” Pet. at 49-50. Petitioner offers no evidence or explanation supporting this speculation. *Id.* Ex. 2001 ¶ 59.

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

Because the Petition merely speculates about what prior art nodes “could” or “may” do, the Petition fails to prove Petitioner’s assertion that the prior art discloses the claim limitation. Ex. 2001 ¶ 60. And because the Petition lacks any allegation that it would have been obvious to modify the prior art to satisfy the claim limitation, Petitioner cannot now fall back on such an obviousness allegation. Petitioner simply failed to meet its initial burden.

C. The Petition does not establish claim 27 is unpatentable.

Claim 27 depends from claim 26 and, thus, is patentable for at least the same reasons that claim 26 is patentable. In addition, the prior art does not disclose the “cluster node module” limitation recited by claim 27. Ex. 2001 ¶ 61.

As explained above, because “cluster node module” is a coined phrase encapsulating the essential features of the cluster node modules disclosed in the specification, the phrase should be construed to mean “a module that cooperates with other cluster node modules to establish intercommunication among nodes in a computer cluster and to exchange messages such that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.” These essential features of the cluster node modules are fundamental parts of the claimed invention of claim 27. Ex. 2001 ¶ 62.

The Petition alleges that the kernel’s “local scheduler” of the prior art is a “cluster node module.” Pet. at 55. However, the Petition does not allege or prove

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

non-standard formats such as “EX-1001,” “Schreiner1,” and “¶216” to make two words count as one word and copying text from the alleged prior art as images on pages 23-24, 27, and 50 of the Petition to avoid quoting the text and counting the words. These instances add up to about 860 excess words. The Board has rejected briefs because they used similar formatting tricks to exceed page or word limits. *See, e.g., Starbucks Corp. v. Ameranth, Inc.*, CBM2015-00091, Paper 16 at 2-3 (P.T.A.B. Jan. 29, 2016) (rejecting a Patent Owner Response that used “at least 28 point” spacing rather than “double spacing”).

V. CONCLUSION

For the foregoing reasons, the Board should deny the Petition.

Respectfully submitted,

KNOBBE, MARTENS, OLSON & BEAR, LLP

Dated: February 10, 2021

By: /Ted M. Cannon/

Jon W. Gurka, Reg. No. 44,139

Ted M. Cannon, Reg. No. 55,036

Cheryl T. Burgess, Reg. No. 55,030

Attorneys for Patent Owner

Advanced Cluster Systems, Inc.

EXHIBIT P

Trials@uspto.gov
571-272-7822

Paper 9
Date: May 5, 2021

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

NVIDIA CORP.,
Petitioner,

v.

ADVANCED CLUSTER SYSTEMS, INC.,
Patent Owner.

IPR2021-00019
Patent 10,333,768 B2

Before KARL D. EASTHOM, ARTHUR M. PESLAK, and
SEAN P. O'HANLON, *Administrative Patent Judges*.

PESLAK, *Administrative Patent Judge*.

DECISION
Denying Institution of *Inter Partes* Review
35 U.S.C. § 314

IPR2021-00019

Patent 10,333,768 B2

(1) a second node

Petitioner contends that “[o]ne of the plurality of nodes comprising a hardware processor discussed above . . . is the ‘second node.’” Pet. 38 (citing Ex. 1005 ¶ 91).

(2) “plurality of processing cores”

Petitioner contends that “Schreiner1 discloses the use of multi-processor systems such as the ‘128 processor SGI Origin 3800 distributed shared memory multiprocessor’ and ‘an 18-processor Sun HPC 6500 system.’” Pet. 38 (citing Ex. 1008, 324–325, 327). Petitioner further contends that “[i]t would have been obvious to implement the system of Schreiner1 using a hardware processor with a plurality of processing cores for one or more of the nodes, including the second node, as taught in the SPARC IV Article and the AMD article.” *Id.*

(3) “the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of the first mathematical expression evaluation to a third node”

Petitioner contends that Schreiner1’s Figure 5 discloses “the second node receives a call from the first node, indicated by task <t,d>, it then executes the mathematical expression indicated by the task call; and the second node then returns a result to the first node in result <t,r>.”

Pet. 41–42 (citing Ex. 1005 ¶ 95; Ex. 1008, 319–320, Fig. 5).

h) “wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression

IPR2021-00019

Patent 10,333,768 B2

evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;”

Patent Owner contends that the Petition fails to establish this limitation because it fails “to show where the prior art teaches the precise order of operations, involving three specific nodes, specified by the claim.” Prelim. Resp. 27. Patent Owner further contends “the Petition refers to the prior art’s general disclosure that multiple nodes may be involved in evaluating mathematical expressions and then speculates that the prior art could work in a manner that would meet the claim limitation.” *Id.* (citing Pet. 44–47; Ex. 2001 ¶ 67).

Petitioner provides three examples to establish that the cited references disclose this limitation. In the first example, Petitioner contends that “Schreiner¹ gives an example where ‘[a]fter the distributed session has been successfully established, two calls of dist[start] create two tasks evaluating the Maple expressions $\text{int}(x^n, x)$ and $\text{int}(x^n, n)$, respectively.’” Pet. 44 (citing Ex. 1008, 310). Petitioner further contends that the “tasks can be sent to one or more of the second, third, and other nodes for evaluation because ‘[t]he execution of a task may take place on any machine connected to the distributed session.’” *Id.* at 44–45 (citing Ex. 1008, 312). Petitioner further contends that the root node *receives results* from the nodes evaluating the mathematical expressions and “*puts them together into a final result* for display to the user” and “the root (first node) waits for the task results to be sent back by executing the wait instructions, and then displays the overall result of the mathematical expression.” *Id.* at 45 (citing Ex. 1005 ¶ 100; Ex. 1008, 310) (emphases added). Petitioner also contends that “Schreiner¹ teaches that task evaluation may depend on other task results” which “means that a third node performing a mathematical expression

IPR2021-00019

Patent 10,333,768 B2

evaluation may need to wait for a result from a second node performing another evaluation.” *Id.* (citing Ex. 1005 ¶ 101; Ex. 1008, 312). We are not persuaded that this contention satisfies the third node limitation for the following reasons.

We agree with Patent Owner that claim 1 requires a precise order of operations involving the recited first, second, and third nodes. The claim recites that the third node is configured to “receive the result of the first mathematical expression evaluation from the second node” with the third node required to “execute at least a second mathematical expression evaluation using the received result and communicate the result of the second mathematical expression evaluation to the first node.” Ex. 1001, 30:53–59. Petitioner specifically contends that the root node “receives results” which it “puts together for display to the user.” Pet. 45. Petitioner’s sequence of events does not satisfy this limitation because Petitioner does not direct us to evidence that the third node will receive a result from the second node, perform a second mathematical evaluation and communicate the result of the second mathematical evaluation to the first node. Petitioner’s attempt to cure this deficiency by claiming that the third node “*may* need to wait for a result from a second node performing another evaluation” does not persuade us because it is based on speculation of what may happen, but more pertinently is contrary to its contention that the results of the second and third node are both sent to the root node. *See* Pet. 45. Consequently, we determine that Petitioner has not sufficiently established that this example discloses the third node limitation.

Petitioner’s second example is based on Figure 5 of Schreiner¹ which is reproduced below:

IPR2021-00019

Patent 10,333,768 B2

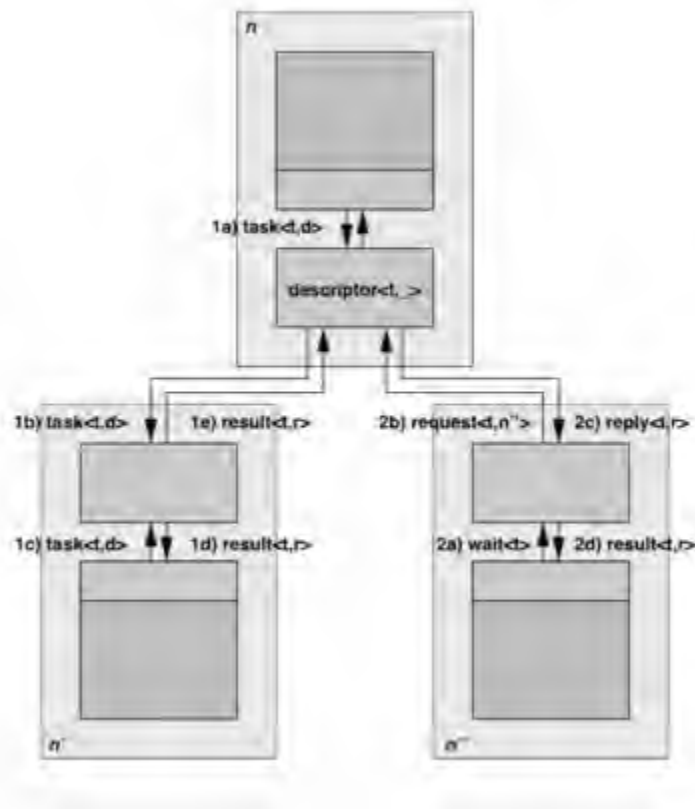


Figure 5 of Schreiner1 is an “execution model” of Distributed Maple.
Ex. 1008, 320.

Petitioner contends that Figure 5 teaches that “the third node (in this case, n'') requests a result from the first node (n), which sends it to the third node after receipt from the second (n').” Pet. 45. Petitioner further contends that the

third node initiates the request through the wait call, indicating that it was evaluating a second expression that was dependent on the result of a first evaluation performed on the second node. A [skilled artisan] would understand that the third node then *could* execute a second mathematical expression using that result, and in fact, that is *likely* why the third node requested the result in the first place. . . . [A] [skilled artisan] would further understand that the third node *could* send the result of the second mathematical expression to the first (root) node if the first node is the node that originally requested that the second mathematical expression be performed, a common case.

IPR2021-00019

Patent 10,333,768 B2

Id. at 46 (citing Ex. 1005 ¶ 102) (emphases added).

We note that this block quotation is repeated in the Petition verbatim from Dr. Tufo’s Declaration. Further, Dr. Tufo does not provide a citation to any portion of Schreiner¹ to support his testimony concerning the execution model shown in Figure 1. Because this testimony is conclusory and not supported by objective evidence, it is entitled to little or no weight. *See Velandier v. Garner*, 348 F.3d 1359, 1371 (Fed. Cir. 2003) (“[W]hat the [PTAB] consistently did was accord little weight to broad conclusory statements that it determined were unsupported by corroborating references. It is within the discretion of the trier of fact to give each item of evidence such weight as it feels appropriate.” (citation omitted)); *see also In re Am. Acad. of Sci. Tech Ctr.*, 367 F.3d 1359, 1368 (Fed. Cir. 2004) (“[T]he [PTAB] is entitled to weigh the declarations and conclude that the lack of factual corroboration warrants discounting the opinions expressed in the declarations” (citations omitted)). Because Petitioner has not provided sufficient objective evidence in support of this example, we determine that Petitioner has not shown that this example discloses the third node limitation.

Petitioner’s third example is based on Figure 1 of Schreiner³ which is reproduced below:

IPR2021-00019

Patent 10,333,768 B2

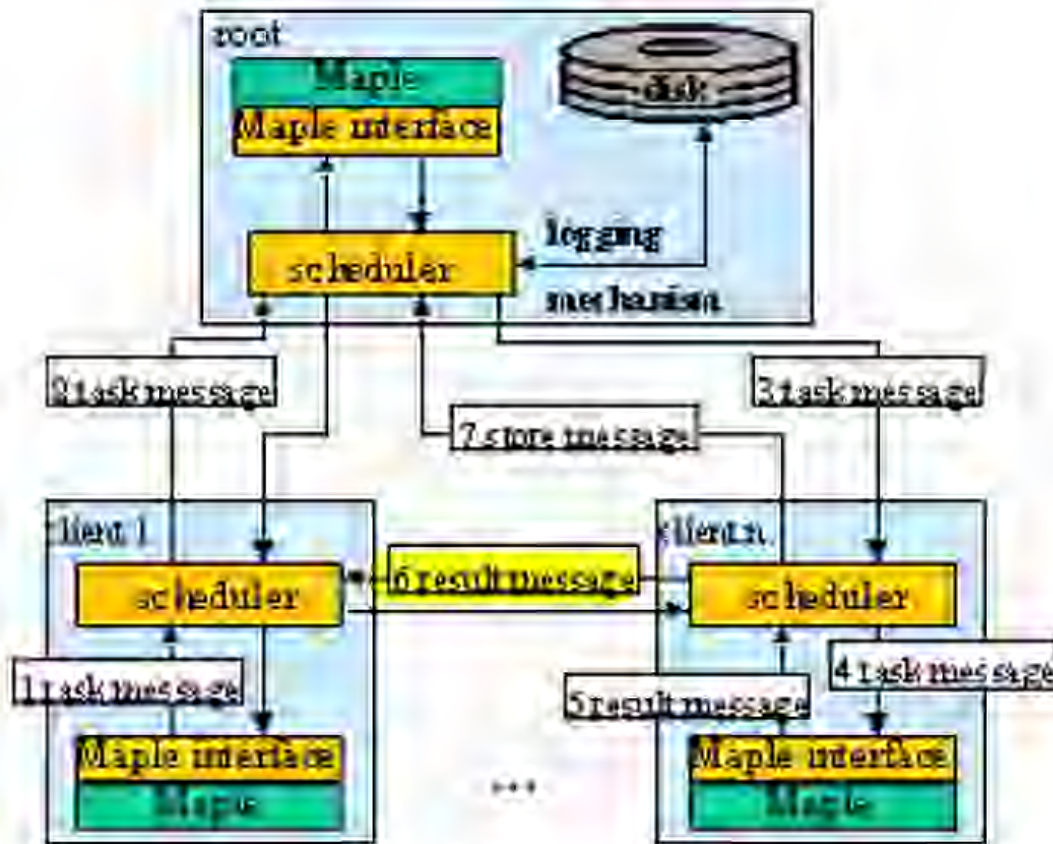


Figure 1 Execution Model

Figure 1 of Schreiner3 is an Execution Model of Distributed Maple.
Ex. 1010, 5.

Petitioner contends that in Figure 1 of Schreiner3, “node ‘client 1’ receives a result for node ‘client n.’” Pet. 47. Petitioner notes that in the case of Figure 1, “the initial tasks originating with the root Maple kernel are not shown.” *Id.* Petitioner further contends that because “the root initiates the mathematical evaluation by distributing tasks to other nodes, node ‘client 1’ is generating subsidiary tasks and evaluating expressions in order to eventually provide results to the root.” *Id.* Petitioner further contends that “once [‘client 1’] receives the result from node ‘client n,’ it may use it to perform at least a second mathematical expression evaluation in order to

IPR2021-00019

Patent 10,333,768 B2

process and return results for tasks sent to it by the root.” *Id.* (citing Ex. 1005 ¶ 103).

We note that this portion of the Petition is taken verbatim from Dr. Tufo’s Declaration. Further, Dr. Tufo does not provide a citation to any portion of any of Schreiner³ to support his testimony concerning the execution model shown in Figure 1 of Schreiner³. The testimony is, thus, entitled to little or no weight because it is conclusory and not based on objective evidence. In addition, neither the Petition nor Dr. Tufo attempt to reconcile the argument that client 1 returns a result to the root node when the box labelled “result message” indicates that result messages are sent between client nodes 1–n, not between the root node and client nodes. Because Petitioner has not provided sufficient evidence in support of this example, we determine that Petitioner has not shown that this example discloses the third node limitation.

Based on the foregoing, we determine that Petitioner fails to establish a reasonable likelihood that claim 1 is unpatentable.

10. Claims 4–10, 18–22, 24, 25, 30, 31, 33, 34

Claims 4–10, 18–22, 24, 25, 30, 31, 33, and 34 depend, directly or indirectly, from independent claim 1. Petitioner provides various contentions and cites additional evidence in connection with these dependent claims. Pet. 52–75. The additional contentions and evidence cited by Petitioner do not cure the deficiencies discussed above in connection with claim 1. We, thus, determine that Petitioner fails to establish a reasonable likelihood that claims 4–10, 18–22, 24, 25, 30, 31, 33, and 34 are unpatentable.

IPR2021-00019

Patent 10,333,768 B2

E. Ground 2: Alleged Obviousness over Schreiner1, Schreiner2, Schreiner3, Distributed Maple Code, Maple Guide, SPARC IV Article, AMD Article, and MPI Standard

Petitioner provides alternate challenges to claims 30, 31, and 34 in this ground. Pet. 10. We have reviewed Petitioner's contentions and additional evidence cited in support of the challenges to claims 30, 31, and 34, and determine that they do not cure the deficiencies discussed above in connection with independent claim 1. *See* Pet. 75–78. Therefore, we determine that Petitioner fails to establish a reasonable likelihood that claims 30, 31, and 34 are unpatentable.

III. CONCLUSION

Because Petitioner has not established a reasonable likelihood that any of the challenged claims are unpatentable, we do not institute *inter partes* review.

IV. ORDER

In consideration of the foregoing, it is hereby
ORDERED that institution of *inter partes* is denied.

EXHIBIT Q

Trials@uspto.gov
571-272-7822

Paper 9
Date: May 5, 2021

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

NVIDIA CORP.,
Petitioner,

v.

ADVANCED CLUSTER SYSTEMS, INC.,
Patent Owner.

IPR2021-00020
Patent 10,333,768 B2

Before KARL D. EASTHOM, ARTHUR M. PESLAK, and
SEAN P. O'HANLON, *Administrative Patent Judges*.

PESLAK, *Administrative Patent Judge*.

DECISION
Denying Institution of *Inter Partes* Review
35 U.S.C. § 314

IPR2021-00020

Patent 10,333,768 B2

- g) *“wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;”*

Patent Owner contends that the Petition fails to establish this limitation because it fails “to show where the prior art teaches the precise order of operations, involving three specific nodes, specified by the claim.” Prelim. Resp. 19. Patent Owner further contends “the Petition refers to the prior art’s general disclosure that multiple nodes may be involved in evaluating mathematical expressions and then speculates that the prior art could work in a manner that would meet the claim limitation.” *Id.* (citing Pet. 44–47; Ex. 2001 ¶ 67).

Petitioner provides two examples to establish that the prior art references disclose this limitation. Pet. 46–50. Petitioner’s first example is based on Figure 5 of Schreiner1, which is reproduced below:

IPR2021-00020

Patent 10,333,768 B2

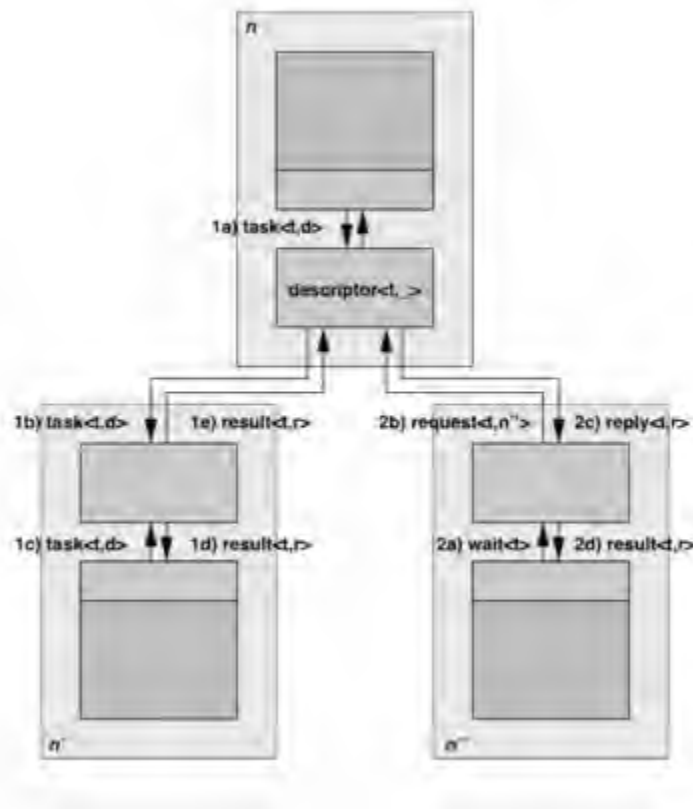


Figure 5 of Schreiner1 is an “execution model” of Distributed Maple.
Ex. 1108, 320.

Petitioner contends that Figure 5 teaches that “the third node (in this case, n'') requests a result from the first node (n), which sends it to the third node after receipt from the second (n').” Pet. 48. Petitioner further contends that the

third node initiates the request through the wait call, indicating that it was evaluating a second expression that was dependent on the result of a first evaluation performed on the second node. A [skilled artisan] would understand that the third node then *could* execute a second mathematical expression using that result, and in fact, that is *likely* why the third node requested the result in the first place. . . . [A] [skilled artisan] would further understand that the third node *could* send the result of the second mathematical expression to the first (root) node if the first node is the node that originally requested that the second mathematical expression be performed, a common case.

IPR2021-00020

Patent 10,333,768 B2

Id. at 48–49 (citing Ex. 1105 ¶ 105) (emphases added).

We note that this block quotation is repeated in the Petition verbatim from Dr. Tufo’s Declaration. Further, Dr. Tufo does not provide a citation to any portion of Schreiner¹ to support his testimony concerning the execution model shown in Figure 1. Because this testimony is conclusory and not supported by objective evidence, it is entitled to little or no weight. *See Velandier v. Garner*, 348 F.3d 1359, 1371 (Fed. Cir. 2003) (“[W]hat the [PTAB] consistently did was accord little weight to broad conclusory statements that it determined were unsupported by corroborating references. It is within the discretion of the trier of fact to give each item of evidence such weight as it feels appropriate.” (citation omitted)); *see also In re Am. Acad. of Sci. Tech Ctr.*, 367 F.3d 1359, 1368 (Fed. Cir. 2004) (“[T]he [PTAB] is entitled to weigh the declarations and conclude that the lack of factual corroboration warrants discounting the opinions expressed in the declarations” (citations omitted)). Because Petitioner has not provided sufficient objective evidence in support of this example, we determine that Petitioner has not shown that this example discloses the third node limitation.

Petitioner’s second example is based on Figure 1 of Schreiner³ which is reproduced below:

IPR2021-00020

Patent 10,333,768 B2

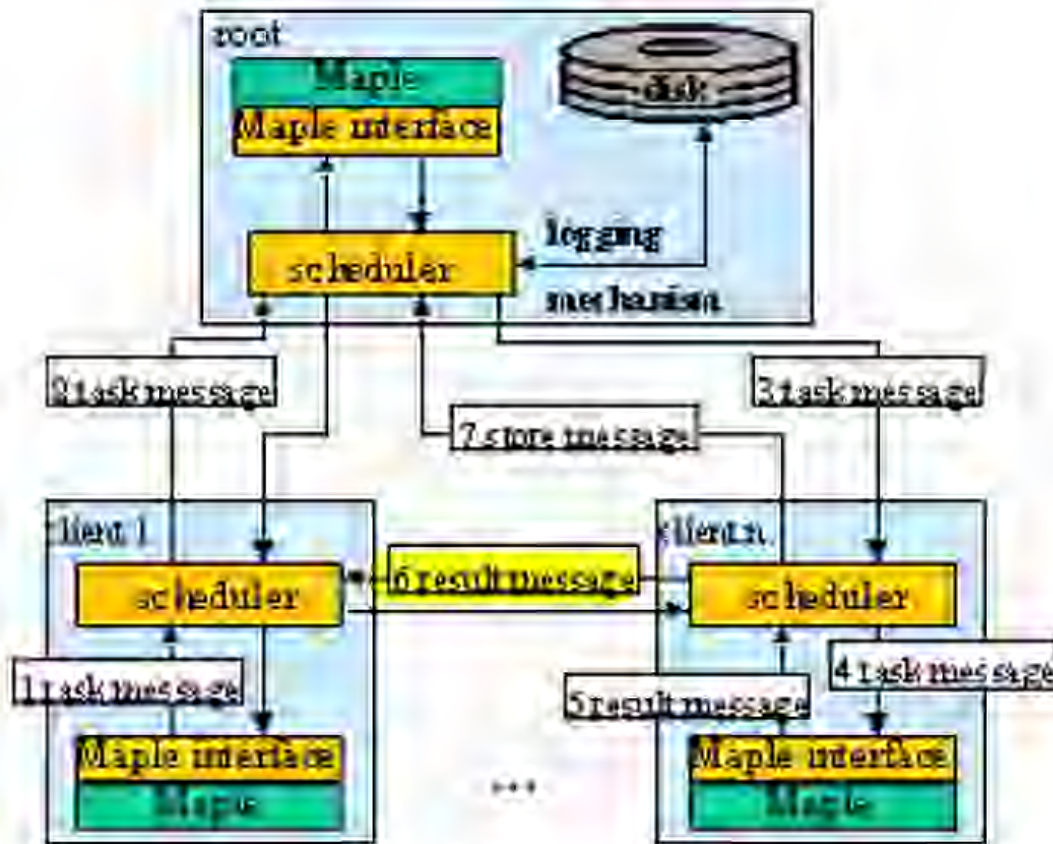


Figure 1 Execution Model

Figure 1 of Schreiner3 is an Execution Model of Distributed Maple.
Ex. 1110, 5.

Petitioner contends that in Figure 1 of Schreiner3, “node ‘client 1’ receives a result from node ‘client n.’” Pet. 49. Petitioner notes that in the case of Figure 1, “the initial tasks originating with the root Maple kernel are not shown.” *Id.* Petitioner further contends that because “the root initiates the mathematical evaluation by distributing tasks to other nodes, node ‘client 1’ is generating subsidiary tasks and evaluating expressions in order to eventually provide results to the root.” *Id.* Petitioner further contends that “once [‘client 1’] receives the result from node ‘client n,’ it may use it to perform at least a second mathematical expression evaluation in order to

IPR2021-00020

Patent 10,333,768 B2

process and return results for tasks sent to it by the root.” *Id.* at 49–50 (citing Ex. 1105 ¶ 111).

We note that this portion of the Petition is taken verbatim from Dr. Tufo’s Declaration. Further, Dr. Tufo does not provide a citation to any portion of any of Schreiner³ to support his testimony concerning the model shown in Figure 1 of Schreiner³. The testimony is, thus, entitled to little or no weight because it is conclusory and not based on objective evidence. In addition, neither the Petition nor Dr. Tufo reconcile the argument that client 1 returns a result to the root node when the box labelled “result message” only indicates that result messages are sent between client nodes 1–n, not between the root node and client nodes. Because Petitioner has not provided sufficient objective evidence in support of this example, we determine that Petitioner has not shown that this example discloses the third node limitation.

Based on the foregoing, we determine that Petitioner fails to establish a reasonable likelihood that claim 26 is unpatentable.

9. Analysis of Claims 27 and 29

Claim 27 depends from claim 26. Ex. 1101, 33:13–23. We have reviewed Petitioner contentions and additional evidence cited in support of the challenge to claim 27, and determine that it does not cure the deficiencies discussed above in connection with claim 26. *See* Pet. 53–56. Therefore, we determine that Petitioner fails to establish a reasonable likelihood that claim 27 is unpatentable.

Independent claim 29 contains the same third node limitation as claim 26. Ex. 1101, 33:38–34:22. Petitioner relies on the same contentions for claim 29 as for claim 26. Pet. 56. For the same reasons discussed above

IPR2021-00020

Patent 10,333,768 B2

with respect to claim 26, we determine that Petitioner fails to establish a reasonable likelihood that claim 29 is unpatentable.

10. Analysis of Claim 35

Claim 35 recites, *inter alia*, “the third node is configured to receive the result of mathematical expression evaluation from the computer cluster node, execute at least a second mathematical expression evaluation using the result of the first mathematical expression evaluation, and communicate a result of the second mathematical expression evaluation to *the first node*.” Ex. 1101, 35:40–36:4 (emphasis added). Petitioner correctly notes that “[t]he term ‘the first node’ lacks antecedent basis.” Pet. 63. Petitioner argues “[t]here are two possible ways to construe the ‘first node’” and contends that Schreiner1 discloses the limitation under either construction. *Id.* at 63–64. Patent Owner does not address the antecedent basis issue, Petitioner’s proposed constructions, or Petitioner’s contentions. *See generally* Prelim. Resp.

Petitioner contends that “[i]f the ‘first node’ is construed as ‘the second node’ (i.e., the root node in Schreiner1), then the ‘third node’ of the claim (e.g., the node in the bottom right of Figure 1 of Schreiner1) is configured to perform this limitation” for the same reasons as for claim 26. Pet. 63. For the same reasons discussed above for claim 26, we are not persuaded that Schreiner1 discloses this limitation.

Petitioner alternately contends that “[i]f the ‘first node’ is construed as ‘the computer cluster node,’ then it would have been obvious to perform a sequence of tasks that result in the third node of the claim receiving a result from the computer cluster node, performing another mathematical expression evaluation, and providing the result of that evaluation to the computer cluster node.” Pet. 63–64. Petitioner further contends that “in the

IPR2021-00020

Patent 10,333,768 B2

normal course of Distributed Maple cluster operation, any node, including the third node may receive results from any node, and may send results to any node.” *Id.* at 64 (citing Ex. 1105 ¶ 143; Ex. 1018, 312). As with Petitioner’s contentions for claim 26, Petitioner and its expert point to no objective evidence but speculate as to what “may” occur.

Based on the foregoing, we determine that Petitioner fails to establish that claim 35 is unpatentable.

11. Analysis of Claim 37

Claim 37 depends from claim 35. Ex. 1101, 36:25–28. We have reviewed Petitioner contentions and additional evidence cited in support of the challenge to claim 37, and determine that it does not cure the deficiencies discussed above in connection with claim 35. *See* Pet. 65. Therefore, we determine that Petitioner fails to establish a reasonable likelihood that claim 37 is unpatentable.

E. Petitioner’s Remaining Obviousness Challenges

Petitioner challenges claims 26, 27, 36, 37, and 39 in these grounds. Pet. 13. We have reviewed Petitioner’s contentions and additional evidence cited in support of the challenges to claims 26, 27, 36, 37, and 39, and determine that they do not cure the deficiencies discussed above in connection with independent claims 26 and 35. *See* Pet. 65–78. Notably, regarding independent claims 26 and 35, Petitioner appears to rely on the same deficient reasoning as discussed above. *Id.* at 66–67. Therefore, we determine that Petitioner fails to establish a reasonable likelihood that claims 26, 27, 36, 37, and 39 are unpatentable.

IPR2021-00020

Patent 10,333,768 B2

III. CONCLUSION

Because Petitioner has not established a reasonable likelihood that any of the challenged claims are unpatentable, we do not institute *inter partes* review.

IV. ORDER

In consideration of the foregoing, it is hereby
ORDERED that institution of *inter partes* is denied.

EXHIBIT R

U.S. Patent No. 10,333,768 (“’768 patent”)

Polymath System

Polymath was a clustered computer system that was known, publicly available, publicly used, offered for sale, sold and described in publications and patent publications in the United States before the earliest alleged priority date of the patents-in-suit, including more than a year prior to the earliest filing date of any application of the patents-in-suit. The Polymath system is embodied in hardware and software components, including but not limited to Apple hardware arranged according to an Appleseed configuration with computer code distributed as Pooch and versions of MacMPI. The Polymath system included multiple versions of the system, including the Polymath 2000 cluster and the subsequent Polymath 2002 cluster.

Polymath was deployed on Power Macintosh hardware and the Macintosh or OS X Operating System and was described as incorporating MacMPI and Pooch in April 2001, in addition to network components (*e.g.*, Ethernet components), and CarbonLib, among other components. *See* ACS_NVIDIA_005621, at _5621 (“a numerically-intensive parallel computing cluster using Power Macintosh hardware and the Macintosh Operating System”), _5622 (“In April 2001, new software debuted that streamlined the user experience ...”), _5622 (“The latest Mac clustering software needs CarbonLib 1.2 or later ...”), _5623 (“Finally, a software package called Pooch is used to operate the cluster ...”) _5624 (“Pooch debuted in April 2001”), _5624 (“MacMPI ... is a wrapper library that calls the Mac OS networking APIs.”). The system used “shrink-wrap applications, such as IDL and Mathematica, to visualize the output of our code. We have even used IDL to display the live output of a plasma physics parallel application” ACS_NVIDIA_005621, at _5628. By 2005, the Pooch software component was described as “a parallel computing and cluster management tool that can organize a job’s files into subdirectories on other nodes and retrieve files on those nodes containing output from completed jobs” in “a wide variety of parallel programming environments ...” ACS_NVIDIA_005607, at _5609. Appleseed implemented as “Mac clusters don’t have a head node, at least not in the sense of the head node found in typical Linux-based clusters. There is no permanent controlling unit or units, so the behavior is much more peer to peer.” ACS_NVIDIA_005607, at _5611. “With your ordered copy of Pooch, you get a one-year subscription, which can be renewed annually at 25% of the purchase price (at the time of renewal) of a new copy.” DR_NVIDIA_00160, at _210. The Polymath system was widely deployed and extensively used more than one year before the earliest alleged priority date of the patents-in-suit.

The Polymath system using the AppleSeed configuration and Pooch qualifies as prior art to U.S. Patent No. 10,333,768 (“’768 patent”) at least under 35 U.S.C. § 102(a) and (b), and alone or with other references, anticipates and/or renders obvious each and every asserted claim, as set forth below. To the extent the Polymath system using the AppleSeed configuration and Pooch does not expressly or inherently disclose one or more limitations of the claims, it would have been obvious to combine the teachings of the Polymath system with the knowledge of one of ordinary skill in the art and with one or more of the references below to render the claims at issue in the ’768 patent invalid. Additional exemplary combinations and reasons for obviousness are described in the cover pleading, and are incorporated by reference herein.

The Polymath system using the AppleSeed configuration and Pooch was designed, developed and implemented by Polymath Research, Inc., including Bedros Afeyan, Kirk Won, Vlad Savanchenko, and others. The Polymath system was known, used, and described in printed publications before the earliest claimed priority date, at least because the work was funded by the Department of Energy under DOE Grant No. DE-FG03-03NA00059. *See, e.g.*, <https://www.osti.gov/servlets/purl/825213> (2002-2004 Technical Report on DOE SSAA Grant # DE-FG03-03NA00059); *see also* Abstract: RP1.00074 : Thomson Scattering Detection of Ponderomotively Driven KEEN Waves in a Laser Produced Plasma (Oct. 24, 2005),

[REDACTED]

<https://meetings.aps.org/Meeting/DPP05/Session/RP1.74>; POLYMATH_0003651 (FW: Our Theory abstracts for APS DPP); POLYMATH_0003655 (ATT151898.htm); POLYMATH_0003836 (Re: Our KEEN Wave related abstracts for APS DPP (6 of them!)); POLYMATH_0003842 (Re: Our KEEN Wave related abstracts for APS DPP (6 of them!)); POLYMATH_0005229 (Kline_APS03_poster_rev2.ppt); POLYMATH_0005253 (Kline_APS03_poster_rev3.ppt); POLYMATH_0005310 (Kline_SSAA_poster.ppt); POLYMATH_0005556 (KEEN waves_APS_2004_rev2.ppt); POLYMATH_0005614 (KEEN waves_ICOPS_2005-2.ppt); POLYMATH_0006299 (Fwd: Management Report for FY 03-04 of DOE SSAA Grant # DE-FG03-03NA00059); POLYMATH_0006301 (Managem_Report_DOE_SSAA_03-.pdf); POLYMATH_0007896 (BBA_CHA04_VNADERBILT.pdf); POLYMATH_0008654 (Management Report for FY 03-04 of DOE SSAA Grant # DE-FG03-03NA00059); POLYMATH_0008655 (Managem_Report_DOE_SSAA_03-04.pdf); POLYMATH_0008768 (DPP_2003_Vlad_parallel_VP_VM.doc); POLYMATH_0008779 (The abstract on parallelization for APS/DPP); POLYMATH_0009022 (Kline_APS03_poster_rev3.ppt); POLYMATH_0009535 (BBA_OMC_SSI_IFSA_2003.doc); POLYMATH_0009536 (Kline_APS03_poster_rev3.ppt); POLYMATH_0009899 (parallel_vp_abstract_DPP03.doc); POLYMATH_0043999 (BBA_ICOPS_MRA_2005_LITE.ppt); POLYMATH_0044036 (BBA_ICOPS_MRA_2005_ICF_SPHERE_LITE.ppt); POLYMATH_0044081 (BBA_ICOPS_MRA_2005_ICF_DT-ICE_LITE.ppt); POLYMATH_0044165 (ATT00002.bin); POLYMATH_0044172 (Re: Our KEEN Wave related abstracts for APS DPP (6 of them!)); POLYMATH_0044176 (Re: Our KEEN Wave related abstracts for APS DPP (6 of them!)); POLYMATH_0044266 (Fwd: Our Theory abstracts for APS DPP); POLYMATH_0044269 (ATT151898.htm); POLYMATH_0064244 (APS KEEN ABSTRACT A LA TWJ); POLYMATH_0064390 (Exprimantal_KEEN_waves_12-04-04.doc); POLYMATH_0064443 (Post_doc_symposium_Jan_2005_rev2.pdf); POLYMATH_0064503 (Exprimantal_KEEN_waves_2-24-05.doc); POLYMATH_0064540 (Exprimantal_KEEN_waves_2-24-05.doc); POLYMATH_0064639 (Kline_aps05_KEEN_Abstract.doc); POLYMATH_0064724 (Re: APS KEEN ABSTRACT A LA TWJ); POLYMATH_0064803 (BBA_OMC_SSI_IFSA_2003_LONG.pdf); POLYMATH_0064819 (KEEN_ARXIV_IFSA+.doc); POLYMATH_0064829 (KEEN_ARXIV_IFSA+.doc); POLYMATH_0064839 (KEEN_ARXIV_IFSA+.doc); POLYMATH_0064849 (KEEN_ARXIV_IFSA+.doc); POLYMATH_0064859 (KEEN_ARXIV_IFSA+.doc); POLYMATH_0065027 (Kline_KEEN_RSI_2-24-05.doc); POLYMATH_0065077 (BBA_OMC_SSI_IFSA_2003.pdf); POLYMATH_0065081 (Kline_RSI_Trident_KEEN.pdf); POLYMATH_0123769 (BBA_OMC_SSI_AA_2003.ps); POLYMATH_0123770 (BBA_OMC_SSI_AA_2003.tex); POLYMATH_0123771 (BBA_OMC_SSI_AA_2003.tex~); POLYMATH_0123785 (DPP_2003_Kline_KEEN_Exp_Design.doc); POLYMATH_0123793 (DPP_2003_Vlad_parallel_VP_VM.doc); POLYMATH_0123801 (BBA_IFSA_KEEN_2003.ps); POLYMATH_0123802 (BBA_IFSA_KEEN_2003.tex); POLYMATH_0123803 (BBA_IFSA_KEEN_2003.tex~); POLYMATH_0123811 (BBA_IFSA_OMC_SSI_2003.ps); POLYMATH_0123812 (BBA_IFSA_OMC_SSI_2003.tex); POLYMATH_0123813 (BBA_IFSA_OMC_SSI_2003.tex~); POLYMATH_0123852 (BBA_CHA04_VNADERBILT.pdf); POLYMATH_0123853 (BBA_CHA04_VNADERBILT.tex); POLYMATH_0123856 (BBA_Mutiscale_Geom_Anal_Fayetteville.pdf); POLYMATH_0123857 (BBA_Mutiscale_Geom_Anal_Fayetteville.tex); POLYMATH_0123878 (Kline_AA05_Abstract.pdf); POLYMATH_0123884 (Kline_aps05_KEEN_Abstract.doc); POLYMATH_0123889 (Morrison abstract APS DPP05.pdf); POLYMATH_0124492 (BBA_KEEN_IFSA_2003_ARCHIVED.doc); POLYMATH_0124498 (BBA_KEEN_IFSA_2003_ArXive.doc); POLYMATH_0124511 (BBA_KEEN_IFSA_2003_FINAL.doc); POLYMATH_0124523 (BBA_KEEN_IFSA_2003_OLD.doc); POLYMATH_0124529 (BBA_OMC_SSI_IFSA_2003_LONG.doc); POLYMATH_0124535 (BBA_OMC_SSI_IFSA_2003_LONG.pdf); POLYMATH_0124540 (BBA_OMC_SSI_IFSA_2003.doc); POLYMATH_0124545 (BBA_OMC_SSI_IFSA_2003.pdf); POLYMATH_0124584 (KEEN_ARXIV_IFSA+.doc); POLYMATH_0124593 (KEEN_ARXIV_IFSA+.pdf); POLYMATH_0124601 (KEEN_ARXIV_IFSA+Stupid.doc); POLYMATH_0124610 (KEEN_IFSA_LATEST.doc); POLYMATH_0124619 (KEEN_IFSA_LATEST.pdf); POLYMATH_0124711

(Kline_KEEN_RSI_2-24-05.doc); POLYMATH_0124741 (Kline_RSI_KEEN_waves.doc); POLYMATH_0126689 (Kline_RSI_Trident_KEEN.pdf); POLYMATH_0126886 (9-SupportingPublications.pdf); POLYMATH_0128962 (Managem Report DOE SSAA 03-04.pdf); POLYMATH_0128967 (Management_Report_03-04.doc); POLYMATH_0128972 (PRI-Scientific_Report_03-04.pdf); POLYMATH_0128993 (Scientific_Report_03-04.doc); POLYMATH_0133660 (KEEN waves_ICOPS_2005-2.ppt); POLYMATH_0133686 (KEEN waves_ICOPS_2005.ppt); POLYMATH_0133711 (Kline_ICOPS_2005_Abstract.doc); POLYMATH_0134026 (BBA_CEA_MRA_2005.ppt); POLYMATH_0134183 (BBA_SPARS05_MRA_2005.pdf); POLYMATH_0134307 (BBA_SPARS05_MRA_2005.ppt); POLYMATH_0134431 (BBA_SPARS05_MRA_2005_Title.pdf); POLYMATH_0014424.

The source code for the same was sent to numerous people, including Dean Dager, who personally tested it on his own AppleSeed-Pooch cluster. *See, e.g.,* s POLYMATH_0001822 (Re: BEPS restarting); POLYMATH_0001826 (Re: Pooch MacMPI window); POLYMATH_0001827 (Re: Pooch connection problem); POLYMATH_0001828 (Re: can not start Pooch on the cluster); POLYMATH_0001830 (Re: can not start Pooch on the cluster); POLYMATH_0001831 (Re: seeming restriction on the array size during compilation and run time); POLYMATH_0001832 (Re: can not start Pooch on the cluster); POLYMATH_0001833 (Re: Pooch upgrade from 18 nodes to 36 nodes); POLYMATH_0001834 (Re: Pooch upgrade from 18 nodes to 36 nodes); POLYMATH_0001835 (Fwd: Pooch upgrade from 18 nodes to 36 nodes); POLYMATH_0001838 (Re: Pooch upgrade from 18 nodes to 36 nodes); POLYMATH_0001839 (QPIC); POLYMATH_0001863 (Re: Pooch license); POLYMATH_0001864 (Re: Pooch license); POLYMATH_0001872 (pic vs. vlasov); POLYMATH_0001893 (Re: Fwd: Beps1 program); POLYMATH_0001899 (Re: Compiling AppleSeed MacMPIf77 and MacMPI_X); POLYMATH_0001900 (Re: Compiling AppleSeed MacMPIf77 and MacMPI_X); POLYMATH_0002100 (Re: mpif90); POLYMATH_0002101 (Re: mpif90); POLYMATH_0002102 (Pooch for Polymath); POLYMATH_0002106 (Re: Pooch on old cluster); POLYMATH_0002107 (Re: XServer + MPI and new MacMPI_S.c); POLYMATH_0002109 (Re: XServer with MPI; MPI + MATLAB); POLYMATH_0002111 (Re: XServer with MPI; MPI + MATLAB); POLYMATH_0002113 (Re: XServer with MPI; MPI + MATLAB); POLYMATH_0002115 (Re: XServer with MPI; MPI + MATLAB); POLYMATH_0002117 (Re: XServer with MPI; MPI + MATLAB); POLYMATH_0002118 (Re: One Pooch Special); POLYMATH_0002120 (One Pooch Special); POLYMATH_0002122 (Re: MPI_BARRIER time outs); POLYMATH_0002123 (g5 and gigabit ethernet); POLYMATH_0002124 (Re: -Rb option); POLYMATH_0002126 (Re: OS I use); POLYMATH_0002128 (Re: OS I use); POLYMATH_0002130 (Re: OS I use); POLYMATH_0002132 (Re: Receive time outs, new comm pattern); POLYMATH_0002133 (Re: Receive time outs, new comm pattern); POLYMATH_0002141 (Re: Receive time out); POLYMATH_0002142 (Re: Receive time out); POLYMATH_0002143 (Re: MPI_WAIT request parameter; Pooch restart); POLYMATH_0002144 (Re: MPI_WAIT request parameter; Pooch restart); POLYMATH_0002146 (Re: transposition routine mysterious problems of large numbers (like 4) remain!); POLYMATH_0002147 (Re: MPI debugger); POLYMATH_0002148 (Re: MPE logging); POLYMATH_0002152 (Re: Pooch command line); POLYMATH_0002153 (Re: xserve g5??); POLYMATH_0002154 (Re: [Fwd: more g5 results]); POLYMATH_0002157 (Re: [Fwd: more g5 results]); POLYMATH_0002158 ([Fwd: more g5 results]); POLYMATH_0002159 (more g5 results); POLYMATH_0002161 (Re: New .C Wrapper and .c sources); POLYMATH_0002162 (Re: Issues about the Pooch command line); POLYMATH_0002163 (Re: Issues about the Pooch command line); POLYMATH_0002164 (Re: Issues about the Pooch command line); POLYMATH_0002168 (G5 Benchmarks); POLYMATH_0002171 (Re: Pooch demo problems); POLYMATH_0002173 (Re: Pooch demo problems); POLYMATH_0002174 (Re: MPI memory problem); POLYMATH_0002177 (Re: MPI memory problem); POLYMATH_0002179 (Re: MPI memory problem); POLYMATH_0002181 (Re: MPI memory problem); POLYMATH_0002194 (Re: matrix transposition, MPI_IRECV/MPI_SEND issue); POLYMATH_0002195 (Re: MPI send/receive problem); POLYMATH_0002199 (Re: MPI send/receive problem); POLYMATH_0002202 (Re: MPI send/receive problem); POLYMATH_0002203 (Re:

[REDACTED]

AppleSeed); POLYMATH_0002204 (Re: MPI); POLYMATH_0002205 (Re: Fwd: MPI); POLYMATH_0002208 (Re: Fwd: Pathetic SPEC2000 numbers for Apple G4); POLYMATH_0003513 (Re: MPI); POLYMATH_0003514 (Re: AppleSeed); POLYMATH_0003515 (Re: MPI send/receive problem); POLYMATH_0003516 (Re: MPI send/receive problem); POLYMATH_0003519 (Re: MPI send/receive problem); POLYMATH_0003523 (Re: matrix transposition, MPI_Irecv/MPI_SEND issue); POLYMATH_0003524 (Re: MPI memory problem); POLYMATH_0003526 (G5 Benchmarks); POLYMATH_0003527 (Re: Issues about the Pooch command line); POLYMATH_0003528 (Re: Issues about the Pooch command line); POLYMATH_0003529 (Re: Issues about the Pooch command line); POLYMATH_0003530 (Re: xserve g5??); POLYMATH_0003531 (Re: Pooch command line); POLYMATH_0003532 (Re: MPI debugger); POLYMATH_0003533 (Re: MPI_WAIT request parameter; Pooch restart); POLYMATH_0003535 (Re: MPI_WAIT request parameter; Pooch restart); POLYMATH_0003536 (Re: Receive time out); POLYMATH_0003537 (Re: Receive time out); POLYMATH_0003541 (Re: Receive time outs, new comm pattern); POLYMATH_0003544 (LAM MPI); POLYMATH_0003545 (Re: OS X LAM install script...); POLYMATH_0003546 (Re: Receive time outs, new comm pattern); POLYMATH_0003547 (Re: OS I use); POLYMATH_0003549 (Re: dual vs single proc machines); POLYMATH_0003550 (Re: successful run on G5 OS 10.3); POLYMATH_0003552 (Re: One Pooch Special); POLYMATH_0003554 (Re: XServer with MPI; MPI + MATLAB); POLYMATH_0003555 (Re: Pooch on old cluster); POLYMATH_0003556 (Pooch for Polymath); POLYMATH_0003559 (Re: mpif90); POLYMATH_0003560 (Re: Compiling AppleSeed MacMPIf77 and MacMPI_X); POLYMATH_0003566 (Re: Compiling AppleSeed MacMPIf77 and MacMPI_X); POLYMATH_0003569 (Re: Pooch license); POLYMATH_0003572 (Re: Pooch license); POLYMATH_0003573 (QPIC); POLYMATH_0013077 (MPI debugger); POLYMATH_0013103 (MPI_WAIT request parameter; Pooch restart); POLYMATH_0013122 (Receive time out); POLYMATH_0013130 (Re: Receive time out); POLYMATH_0013159 (Receive time outs, new comm pattern); POLYMATH_0013164 (Re: Receive time outs, new comm pattern); POLYMATH_0013213 (dual vs single proc machines); POLYMATH_0013215 (successful run on G5 OS 10.3); POLYMATH_0013218 (XServer with MPI; MPI + MATLAB); POLYMATH_0013231 (Pooch on old cluster); POLYMATH_0016863 (mpif90); POLYMATH_0027581 (Pooch license); POLYMATH_0040010 (Pooch and MPICH); POLYMATH_0040023 (Re: Pooch and MPICH); POLYMATH_0040467 (Running MPI code on Pooch); POLYMATH_0040783 (Pooch upgrade from 18 nodes to 36 nodes); POLYMATH_0040785 (Re: Pooch upgrade from 18 nodes to 36 nodes); POLYMATH_0040828 (Re: Pooch upgrade from 18 nodes to 36 nodes); POLYMATH_0040834 (Fwd: Pooch upgrade from 18 nodes to 36 nodes); POLYMATH_0040943 (Re: Pooch upgrade from 18 nodes to 36 nodes); POLYMATH_0040945 (Re: Pooch upgrade from 18 nodes to 36 nodes); POLYMATH_0043130 (Re: One Pooch Special); POLYMATH_0043971 (Re: Pooch license); POLYMATH_0043972 (Re: Pooch license); POLYMATH_0044651 (Re: QPIC); POLYMATH_0045526 (Re: Pooch upgrade from 18 nodes to 36 nodes); POLYMATH_0045551 (Re: Pooch upgrade from 18 nodes to 36 nodes); POLYMATH_0045552 (Re: Pooch upgrade from 18 nodes to 36 nodes); POLYMATH_0045611 (Re: Pooch upgrade from 18 nodes to 36 nodes); POLYMATH_0045764 (Re: Pooch upgrade from 18 nodes to 36 nodes).

APPLESEED-POOCH, MPI, AND CLUSTER COMPUTING PRIOR ART REFERENCES

The design, operation, public availability, public use and offer for sale and sale of the Polymath system are described in the following references (collectively, the “AppleSeed-Pooch references”), which are obvious to combine at least because they collectively describe the Polymath system using the AppleSeed configuration and Pooch and related technology. Furthermore, to the extent the Polymath system using the AppleSeed configuration and Pooch does not disclose one or more limitations of the claims, it would have been obvious to combine the teachings of Polymath with the knowledge of one of ordinary skill in the art and with one or more of the references below to render the claims at issue in the ’768 patent invalid. The following exemplary references disclose the MPI protocol and were obvious to combine with any of the other Polymath system documentation at least because



the Polymath system was touted as operating with MPI motivating a POSITA to adapt a large number of available MPI-based computational libraries, *e.g.*, FFTW, to the Polymath system using the AppleSeed configuration and Pooch without undue experimentation:

Citation	Publication Date	Bates Number
D. E. Dauger & V. K. Decyk, <i>Plug-and-play cluster computing: high-performance computing for the mainstream</i> , COMPUTING IN SCIENCE & ENGINEERING, vol. 7, no. 2, pp. 27-33, (March 2005)	February 1, 2005	ACS_NVIDIA_005607
D. E. Dauger & V. K. Decyk, <i>Plug-and-Play” Cluster Computing: HPC Designed for the Mainstream Scientist</i> , IEEE ISSC, LECTURE NOTES IN COMPUTER SCIENCE, 3515:84–90 (2005)	May 25, 2005	ACS_NVIDIA_005614
D. E. Dauger & Viktor D. Decyk, <i>“Plug-and-Play” Cluster Computing using Mac OS X</i> , IEEE INTERNATIONAL CONFERENCE ON CLUSTER COMPUTING, 430–435 (2003)	January 8, 2004	ACS_NVIDIA_005715
D. E. Dauger & V. K. Decyk, <i>Numerically-Intensive “Plug-and-Play” Parallel Computing</i> , PROCEEDINGS 2001 IEEE INTERNATIONAL CONFERENCE ON CLUSTER COMPUTING, LAS VEGAS, NEVADA, USA, 75–82 (October 14-17, 2001)	October 14, 2001	ACS_NVIDIA_005621
Decyk Viktor K & Dauger Dean E, <i>Appleseed: A Parallel Macintosh Cluster for Scientific Computing</i> , JOURNAL PLASMA FUSION RESEARCH, Vol. 79, No. 8 (August 2003), p. 722-779	2003	ACS_NVIDIA_005613
V. K. Decyk, D. E. Dauger, & Pieter R. Kokelear, <i>How to Build an AppleSeed: A Parallel Macintosh Cluster for Numerically Intensive Computing</i> , PHYSICA SCRIPTA, 84 (2000), p. 85	2000	ACS_NVIDIA_005662; ACS_NVIDIA_005680
V. K. Decyk & D. E. Dauger, <i>Supercomputing for the Masses: A Parallel Macintosh Cluster, Parallel Processing and Applied Mathematics</i> , Springer (2002), pp. 10-22	2002	NVIDIA-ACS-0166308
V. K. Decyk, <i>Skeleton PIC code for parallel computers</i> , COMPUTER PHYSICS COMMUNICATIONS 87 (1995), pp. 85-94	1995	NVIDIA-ACS-0165933
V. K. Decyk, D. E. Dauger, & Pieter R. Kokelaar, <i>Appleseed: A Parallel Macintosh Cluster for Numerically Intensive Computing</i> , COMPUTER PHYSICS COMMUNICATIONS, 121: 627 (1999)	1999	ACS_NVIDIA_005692; ACS_NVIDIA_005700
Dean Dauger, <i>Method and System for Parallel Operation and Control of Legacy Computer Clusters</i> , US Patent Application Publication 2003/0195931 (“the ‘931 Pub”)	October 16, 2003	NVIDIA-ACS-0166349
Dean Edward Dauger, <i>Semiclassical Modeling of Quantum-Mechanical Multiparticle Systems using Parallel Particle-In-Cell Methods</i> , UCLA Dissertation (2001)	2001	DR_NVIDIA_002082
D. E. Dauger, <i>Pooch Parallel Operation and Control Heuristic Application Manual</i> , v1.4	January 6, 2004	DR_NVIDIA_000160
D. E. Dauger, <i>Pooch Parallel Operation and Control Heuristic Application Manual</i> v1.5	June 28, 2004	DR_NVIDIA_001448; NVIDIA-ACS-0166377
Charles W. Moore, <i>The AppleSeed Project: Clustered Power Macs Outperform Cray Supercomputer</i> , LOW END MAC (Apr. 19, 2000)	April 19, 2000	NVIDIA-ACS-0165985

Citation	Publication Date	Bates Number
D. E. Dauger, <i>"Plug-and-Play" Clustering Build your cluster in minutes</i> , APPLE HPC SEMINAR SERIES (July 1, 2005)	July 1, 2005	ACS_NVIDIA_001238
Leander Kahney, <i>That's a Whole Lot of Power, Mac</i> , WIRED (Jan. 29, 2002)	January 29, 2002	NVIDIA-ACS-0166332; NVIDIA-ACS-0166333
Dean Dauger, <i>READ ME for The Launch Den Mother and Launch Puppy</i>	1999	DR_NVIDIA_000703
Dean Dauger, <i>Advanced Information for The Launch Den Mother and Launch Puppy</i>	March 2000	DR_NVIDIA_000713
Computer source code for MacMPI, Pooch, and related functions and libraries including but not limited to modules and related files associated with the Pooch SDK, including but not limited to all versions of MacMPI_X.f, MacMPI_X.c, mpi.h, mpif.h, MacMPIlg77.c, MacMPI_S.c, MacMPIlcf.c, MacMPIlf77.c, ParallelFractalEngine.c, ParallelFractalEngine.h, ParallelFractalMain.c, knock.c, knock.f, paralleladder.c, parallelpascalstriangle.c, pingpong.c, pingpong.f.	2002	
D. E. Dauger, V. K. Decyk, J. W. Tonge, <i>Using semiclassical trajectories for the time-evolution of interacting quantum-mechanical systems</i> , 209 J. OF COMPUTATIONAL PHYSICS 559 (May 23, 2005)	May 23, 2005	NVIDIA-ACS-1137663; see also NVIDIA-ACS-0166510
Stefan Goedeckera, Mireille Bouletb, Thierry Deutsch, <i>An efficient 3-dim FFT for plane wave electronic structure calculations on massively parallel machines composed of multiprocessor nodes</i> , COMPUTER PHYSICS COMMUNICATIONS 154 (2003)	March 31, 2003	NVIDIA-ACS-1380013
<i>Dauger Research Profiled on National Television Part 1</i>	May 3, 2005	NVIDIA-ACS-1137689
<i>Dauger Research Profiled on National Television Part 2</i>	May 3, 2005	NVIDIA-ACS-1137688
<i>Dauger Research Profiled on National Television Part 3</i>	May 3, 2005	NVIDIA-ACS-1137687
<i>Apple WWDC 2004 Session 642</i>	2002	NVIDIA-ACS-1137686
<i>Apple Canada Video Introduction</i>	2002	NVIDIA-ACS-1122559
<i>Apple Canada Video Why Parallel</i>	2002	NVIDIA-ACS-1122562
<i>Apple Canada Video Why Macs</i>	2002	NVIDIA-ACS-1122561
<i>Apple Canada Video Mac Recipe</i>	2002	NVIDIA-ACS-1122560
<i>Apple Canada Video Demo Pt 1</i>	2002	NVIDIA-ACS-1122554
<i>Apple Canada Video Demo Pt 2</i>	2002	NVIDIA-ACS-1122555
<i>Apple Canada Video Depo PT 3</i>	2002	NVIDIA-ACS-1122556

Citation	Publication Date	Bates Number
<i>Apple Canada Video Demo PT 4</i>	2002	NVIDIA-ACS-1122557
<i>Apple Canada Video Demo Pt 5</i>	2002	NVIDIA-ACS-1122558
<i>Apple Canada Video Pooch 4 User Interfaces</i>	2002	NVIDIA-ACS-1122553
1999.04.28 - Appleseed Development Page	1999	NVIDIA-ACS-1332442
2000.06.22 - Appleseed Development Page	2000	NVIDIA-ACS-1332451
2000.08.16 - AppleSeed Recipe	2000	NVIDIA-ACS-1332455
2001.02.01 - Appleseed Development Page	2001	NVIDIA-ACS-1332514
2001.04.29 - Appleseed Development Page	2001	NVIDIA-ACS-1332522
2001.05.01 - AppleSeed Recipe	2001	NVIDIA-ACS-1332526
2001.06.25 - Apple - Science & Technology - UCLA Project AppleSeed	2001	NVIDIA-ACS-1332533
2001.06.25 - Apple - Science & Technology - UCLA Project AppleSeed - Page 2	2001	NVIDIA-ACS-1332531
Other AppleSeed Clusters (Oct. 12, 2004)	2004	NVIDIA-ACS-1136464
Dauger Research, Inc., - Pooch v1.4 Announcement	Jan. 6, 2004	NVIDIA-ACS-1133207
LAM_MPI with Pooch	Oct. 20, 2004	NVIDIA-ACS-1111696
mpich with Pooch	Dec. 16, 2004	NVIDIA-ACS-1111700
Pooch - Mac Cluster Recipe	Dec. 14, 2004	NVIDIA-ACS-1111703
Pooch - Mac Cluster Recipe	Jan. 18, 2006	NVIDIA-ACS-1111705
Pooch - MacMPI_X Visualization	Dec. 13, 2004	NVIDIA-ACS-1111707
Pooch - MPI Implementation	Dec. 28, 2005	NVIDIA-ACS-1111709
Pooch - Other Links	Dec. 14, 2004	NVIDIA-ACS-1111710
Pooch - Parallel Adder	Oct. 19, 2004	NVIDIA-ACS-1111712
Pooch - Parallel Circle Pi	Apr. 4, 2005	NVIDIA-ACS-1111720
Pooch - Parallel Knock	Oct. 19, 2004	NVIDIA-ACS-1111735
Pooch - Parallel Life	Apr. 4, 2005	NVIDIA-ACS-1111740

Citation	Publication Date	Bates Number
Pooch - Parallel Pascal's Triangle	Oct. 28, 2004	NVIDIA-ACS-1111756
Pooch - Parallel Zoology	Oct. 28, 2004	NVIDIA-ACS-1111767
Pooch - Purchase	Dec. 29, 2005	NVIDIA-ACS-1111776
Pooch Apple Canada Video Page	Dec. 20, 2004	NVIDIA-ACS-1111777
Pooch Command Line Interface	Feb. 17, 2006	NVIDIA-ACS-1111779
Pooch FAQ Page	Oct. 11, 2004	NVIDIA-ACS-1111780
Pooch Parallel Applications Page	Dec. 16, 2004	NVIDIA-ACS-1136845
Pooch Pro 1.6 Download	Dec. 28, 2005	NVIDIA-ACS-1111787
Pooch Pro v1.6 Release	10-May-05	NVIDIA-ACS-1111788
Pooch Scripting and Automation	Feb. 17, 2006	NVIDIA-ACS-1111790
Pooch Users	Dec. 5, 2004	NVIDIA-ACS-1136847
Apple, <i>Inside Macintosh - Apple Numerics Manual</i> (2nd Ed 1988)	1988	NVIDIA-ACS-1325711
Apple, <i>Inside Macintosh - Designing Cards and Drivers</i> (2nd Ed 1990)	1990	NVIDIA-ACS-1326010
Apple, <i>Inside Macintosh - Designing Cards and Drivers</i> (3rd Ed 1992)	1992	NVIDIA-ACS-1326483
Apple, <i>Inside Macintosh - Designing Cards and Drivers for the Macintosh II and SE</i> (1987)	1987	NVIDIA-ACS-1327156
Apple, <i>Inside Macintosh - Guide to Macintosh Family Hardware</i> (2nd Ed 1990)	1990	NVIDIA-ACS-1327425
Apple, <i>Inside Macintosh - PowerPC Numerics</i> (1994)	1994	NVIDIA-ACS-1327984
Apple, <i>Inside Macintosh - PowerPC System Software</i> (1994)	1994	NVIDIA-ACS-1328316
Apple, <i>Inside Macintosh - Processes</i> (1994)	1994	NVIDIA-ACS-1328534
Apple, <i>Inside Macintosh - Sound</i> (1994)	1994	NVIDIA-ACS-1328740
Apple, <i>Inside Macintosh - Text</i> (1993)	1993	NVIDIA-ACS-1329274
Apple, <i>Inside Macintosh: Interapplication Communication</i> (1993)	1993	NVIDIA-ACS-1134268
Apple, <i>Inside Macintosh - AOCE Application Interfaces</i> (1994)	1994	NVIDIA-ACS-1314219
Apple, <i>Inside Macintosh - AOCE Service Access Modules</i> (1994)	1994	NVIDIA-ACS-1315589

Citation	Publication Date	Bates Number
Apple, <i>Inside Macintosh - Devices</i> (1994)	1994	NVIDIA-ACS-1316063
Apple, <i>Inside Macintosh - Files</i> (1992)	1992	NVIDIA-ACS-1316607
Apple, <i>Inside Macintosh - Human Interface Guidelines</i> (1992)	1992	NVIDIA-ACS-1317143
Apple, <i>Inside Macintosh - Imaging With QuickDraw</i> (1994)	1994	NVIDIA-ACS-1317553
Apple, <i>Inside Macintosh - Macintosh Toolbox Essentials</i> (1992)	1992	NVIDIA-ACS-1318377
Apple, <i>Inside Macintosh - Memory</i> (1992)	1992	NVIDIA-ACS-1319286
Apple, <i>Inside Macintosh - More Macintosh Toolbox</i> (1993)	1993	NVIDIA-ACS-1319590
Apple, <i>Inside Macintosh - Networking</i> (1994)	1994	NVIDIA-ACS-1320503
Apple, <i>Inside Macintosh - Operating System Utilities</i> (1994)	1994	NVIDIA-ACS-1321080
Apple, <i>Inside Macintosh - Overview</i> (1992)	1992	NVIDIA-ACS-1321452
Apple, <i>Inside Macintosh - Volume I</i> (1985)	1985	NVIDIA-ACS-1321720
Apple, <i>Inside Macintosh - Volume II</i> (1985)	1985	NVIDIA-ACS-1322280
Apple, <i>Inside Macintosh - Volume III</i> (1985)	1985	NVIDIA-ACS-1322717
Apple, <i>Inside Macintosh - Volume IV</i> (1986)	1986	NVIDIA-ACS-1323004
Apple, <i>Inside Macintosh - Volume V</i> (1986)	1986	NVIDIA-ACS-1323338
Apple, <i>Inside Macintosh - Volume VI</i> (1991)	1991	NVIDIA-ACS-1323962
Apple, <i>AppleTalk Remote Access Users Guide</i> (1991)	1991	NVIDIA-ACS-1330492
Apple, <i>Inside AppleTalk</i> (2nd Ed. 1990)	1990	NVIDIA-ACS-1331064
Apple, <i>Inside Macintosh - Networking with Open Transport</i> (1997)	1997	NVIDIA-ACS-1381446
Apple, <i>Inside Macintosh - Thread Manager</i> (1999)	1999	NVIDIA-ACS-1382332
Apple, <i>Inside The Macintosh Communications Toolbox</i> (1991)	1991	NVIDIA-ACS-1382440
Apple, <i>Planning and Managing AppleTalk Networks</i> (1991)	1991	NVIDIA-ACS-1377988
Nicole Hemosth, <i>Startup Aims to Bring Parallel Applications to the Masses</i> , HPC WIRE, (Nov. 16, 2011)	November 16, 2011	NVIDIA-ACS-1099785
<i>Visualizing Message Passing with MacMPI</i> (Apr. 4, 2005)	2005	NVIDIA-ACS-1111793

Citation	Publication Date	Bates Number
Apple, <i>Networking the Macintosh a step-by-step guide to using AppleTalk in Business Environments</i> (1993)	1993	NVIDIA-ACS-1331655
Apple, <i>Programming With AppleTalk</i> (1991)	1991	NVIDIA-ACS-1332024
A. Oppenheimer, <i>AppleTalk Update-Based Routing Protocol - Enhanced AppleTalk Routing</i> (Aug. 1993)	Aug. 1993	NVIDIA-ACS-1378289
Mac Cluster Deployments (Dec. 20, 2004)	2004	NVIDIA-ACS-1111698
Dauger Research - Altive Fractal Carbon Demonstration Software (Feb. 12, 2005)	2005	NVIDIA-ACS-1133199
Dauger Research Store (Oct. 15, 2004)	2004	NVIDIA-ACS-1111694
Dauger Research, Inc - Fresnel Diffraction Webpage (Dec. 5, 2004)	2004	NVIDIA-ACS-1133202
1999.02.19 - Launch Den Mother and Launch Puppy	1999	NVIDIA-ACS-1332440
1999.05.04 - Viktor Decyk's UCLA Webpage	1999	NVIDIA-ACS-1332446
1999.05.06 - UCLA Plasma Physics Group Homepage	1999	NVIDIA-ACS-1332449
2000.10 - PowerPC-G4velocityengine	2000	NVIDIA-ACS-1332456
2001.04.29 - Apple - Science & Technology - Scientific Computing	2001	NVIDIA-ACS-1332520
2001.04.29 - Apple - Science & Technology - Scientific Computing - Part 2	2001	NVIDIA-ACS-1332518
2001.06.13 - Apple - Science & Technology - High Performance Computing	2001	NVIDIA-ACS-1332527
2001.06.25 - Apple - Science & Technology - Cluster Construction	2001	NVIDIA-ACS-1332529
2002 Technical Information	2002	NVIDIA-ACS-1333119
2003 – Xserve Cluster Node Quick Start Guide	2003	NVIDIA-ACS-1333130
2003 Xserve Datasheet	2003	NVIDIA-ACS-1333149
2003 Xserve Technology Overview	2003	NVIDIA-ACS-1333154
2003 Xserve Performance Data	2003	NVIDIA-ACS-1333187
2003.02.02 - Apple - Xserve - Performance	2003	NVIDIA-ACS-1333193
2003.04.01 - Apple - Xserve	2003	NVIDIA-ACS-1333208
2003.04.01 - Apple - Xserve - Management	2003	NVIDIA-ACS-1333198
2003.04.01 - Apple - Xserve - Storage	2003	NVIDIA-ACS-1333201

Citation	Publication Date	Bates Number
2003.04.01 - Apple - Xserve - Technical Specifications	2003	NVIDIA-ACS-1333204
2003.04.09 - ADC Developer Documentation	2003	NVIDIA-ACS-1333211
2003.06.15 - Apple - Server - Solutions - Clustering Resources	2003	NVIDIA-ACS-1333215
2003.10.09 - Apple - Server - Solutions - Compute Clustering	2003	NVIDIA-ACS-1333218
1999.10.03 - Apple - Products - Power Mac G4	1999	NVIDIA-ACS-1379927
1999.10.10 - Apple - Products - Power Mac G4 Expansion	1999	NVIDIA-ACS-1379930
1999.10.11 - Apple - Products - Power Mac G4 Tech Specs	1999	NVIDIA-ACS-1379932
1999.10.13 - Apple - Products - Power Mac G4 Graphics Acceleration	1999	NVIDIA-ACS-1379935
1999.10.13 - Apple - Products - Power Mac G4 Processor	1999	NVIDIA-ACS-1379937
2001.01.24 - Apple - Products - Power Mac G4	2001	NVIDIA-ACS-1379941
Power Mac G4 Developer Note (1999)	1999	NVIDIA-ACS-1387380
Power Mac G4 Developer Note (2000)	2000	NVIDIA-ACS-1387469
Power Max G4 Developer Notes (2001)	2001	NVIDIA-ACS-1387558
PowerBookG3Series	2001	NVIDIA-ACS-1385553
PowerMac_G3	2001	NVIDIA-ACS-1385641
PowerMacintosh_G3	2001	NVIDIA-ACS-1385721
Power Book G4 Developer Note (2002)	2002	NVIDIA-ACS-1384540
2002.06.04 - Apple - Xserve - Technical Specifications	2002	NVIDIA-ACS-1380445
2002.08.02 - Apple - Xserve	2002	NVIDIA-ACS-1380448
2002.09.03 - Apple - Xserve - Architecture	2002	NVIDIA-ACS-1380451
2002.09.03 - Apple - Xserve - Storage	2002	NVIDIA-ACS-1380453
2002.09.04 - Apple - Xserve - Design	2002	NVIDIA-ACS-1380455
Mac OS X Server v10.2 Administrator Guide (2002)	2002	NVIDIA-ACS-1380457
<i>MPI: A Message-Passing Interface Standard</i> (June 12, 1995)	June 12, 1995	NVIDIA-ACS-0164777

Citation	Publication Date	Bates Number
<i>MPI: The Message Passing Interface</i> (Nov. 15, 2003)	Nov. 15, 2003	NVIDIA-ACS-1378569
Peter S. Pacheco, <i>A User's Guide to MPI</i> (1998)	1998	NVIDIA-ACS-1133144
M. Snir, S. Otto, S. Huss-Lederman, D. Walker, & J. Dongarra, <i>MPI: The Complete Reference</i> (1st ed. 1996)	1996	NVIDIA-ACS-1135272
M. Snir, S. Otto, S. Huss-Lederman, D. Walker, & J. Dongarra, <i>MPI The Complete Reference</i> , Vol. 1 (2nd ed. 1998)	1998	NVIDIA-ACS-1135622
M. Snir, S. Otto, S. Huss-Lederman, D. Walker, & J. Dongarra, <i>MPI The Complete Reference</i> , Vol. 2 (2nd ed. 1999)	1999	NVIDIA-ACS-1136074
W. Gropp, E. Lusk, & A. Skjellum, <i>Using MPI Portable Parallel Programming</i> (2nd Ed. 1999)	1999	NVIDIA-ACS-1136859
<i>Programming with MPI on clusters</i>	Feb. 2001	NVIDIA-ACS-1137251; NVIDIA-ACS-1330392
Stefan Goedecker & Adolfo Hoisie, <i>Performance Optimization of Numerically Intensive Codes</i> (2001)	2001	NVIDIA-ACS-1380125
<i>Using MPI-2 Advanced Features of the Message-Passing Interface</i>	1999	NVIDIA-ACS-1137251
Gregory F. Pfister, <i>In Search of Clusters</i> (1st Ed. 1995)	1995	NVIDIA-ACS-1133212
Gregory F. Pfister, <i>In Search of Clusters</i> (2nd Ed. 1998)	1998	NVIDIA-ACS-1133656
T. Mattson, B. Sanders, & B. Massingill, <i>Patterns For Parallel Programming</i> (2005)	2005	NVIDIA-ACS-1136471
Hesham El-Rewini & Mostafa Abd-El-Barr, <i>Advanced Computer Architecture And Parallel Processing</i> (2005)	2005	NVIDIA-ACS-1373000
Lloyd D. Fosdick, Elizabeth R. Jessup, Carolyn J. C. Schauble, Gitta Domik, <i>An Introduction To High Performance Scientific Computing</i> (1996)	1996	NVIDIA-ACS-1373284
Thomas Sterling, <i>Beowulf Cluster Computing with Windows</i> (2002)	2002	NVIDIA-ACS-1374073
Harold S. Stone, <i>High Performance Computer Architecture</i> (3rd Ed. 1993)	1993	NVIDIA-ACS-1375273
J. Cownie & W. Gropp, <i>A Standard Interface for Debugger Access to Message Queue Information in MPI</i> , EURO PVM/MPI 1999: RECENT ADVANCES IN PARALLEL VIRTUAL MACHINE AND MESSAGE PASSING INTERFACE (Sept. 1999) ("Cownie 1999")	1999	NVIDIA-ACS-0170449

Citation	Publication Date	Bates Number
Kevin Dowd & Charles Severance, <i>High Performance Computing</i> (2nd Ed. 2005)	2005	NVIDIA-ACS-1375799
Jonathan Schaeffer, <i>High Performance Computing Systems and Applications</i> (1998)	1998	NVIDIA-ACS-1376256
J.J. Dongarra, L. Grandinetti, G.R. Joubert, J. Kowalik, <i>High Performance Computing Technology, Methods and Applications</i> (1995)	1995	NVIDIA-ACS-1376705
<i>High Performance Servers</i> (2nd Ed. 1991)	1991	NVIDIA-ACS-1377142
Timothy G. Mattson, Beverly A. Sanders, Berna L. Massingill, <i>Patterns for Parallel Programming</i> (2005)	2005	NVIDIA-ACS-1377252
David Barkai, <i>Peer-to-Peer Computing: Technologies for Sharing and Collaborating on the Net</i> (2001)	2001	NVIDIA-ACS-1377622
Rajkumar Buyya, <i>High Performance Cluster Computing: Architectures and Systems</i> , Volume 1 (1999)	1999	NVIDIA-ACS-1137690
Rajkumar Buyya, <i>High Performance Cluster Computing: Architectures and Systems</i> , Volume 2 (1999)	1999	NVIDIA-ACS-1138570
European Patent No. 1,229,442 - Peer- To- Peer Computing Architecture	2002	NVIDIA-ACS-1378807
European Patent No. 1,229,443 - Peer- To- Peer Computing Architecture With Groups	2002	NVIDIA-ACS-1378910
U.S. Patent No. 7,065,579 - System Using Peer Discovery And Peer Membership Protocols For Accessing Peer-To-Peer Platform Resources On A Network	2002	NVIDIA-ACS-1379005
U.S. Patent No. 7,136,927 - Peer-To-Peer Resource Resolution	2002	NVIDIA-ACS-1379071
U.S. Patent No. 7,167,920 - Peer-To-Peer Communication Pipes	2002	NVIDIA-ACS-1379137
U.S. Patent No. 7,206,841 - Rendezvous For Locating Peer-To-Peer Resources	2002	NVIDIA-ACS-1379210
U.S. Patent No. 7,340,500 - Providing Peer Groups In A Peer-To-Peer Environment	2002	NVIDIA-ACS-1379288
U.S. Patent No. 7,401,152 - Resource Identifiers For A Peer-To-Peer Environment	2002	NVIDIA-ACS-1379365
U.S. Patent No. 7,401,153 - Peer-To-Peer Computing Architecture	2002	NVIDIA-ACS-1379431
U.S. Patent No. 7,533,172 - Advertisements For Peer-To-Peer Computing Resources	2002	NVIDIA-ACS-1379501
U.S. Patent No. 7,574,523 - Relay Peers For Extending Peer Availability In A Peer-To-Peer Networking Environment	2002	NVIDIA-ACS-1379570

Citation	Publication Date	Bates Number
U.S. Patent No. 8,160,077 - Peer-To-Peer Communication Pipes	2002	NVIDIA-ACS-1379638
U.S. Patent No. 8,176,189 - Peer-To-Peer Network Computing Platform	2002	NVIDIA-ACS-1379702
U.S. Patent No. 8,359,397 - Reliable Peer-To-Peer Connections	2002	NVIDIA-ACS-1379764
WO2002-057917 - Peer-To-Peer Network Computing Platform	2002	NVIDIA-ACS-1379837
Paul Swarztrauber, <i>Multiprocessor FFTs</i> (1987)	1987	NVIDIA-ACS-1380104
Parallel Computer Architecture A Hardware Software Approach (1998)	1998	NVIDIA-ACS-0346702

VLASOV-MAXWELL AND VLASOV-POISSON SYSTEM OF EQUATIONS PRIOR ART REFERENCES

The following exemplary references disclose the Vlasov-Maxwell and Vlasov-Poisson system of equations, and other mathematical programs and equations (e.g., FFTW), including methods of executing parallel versions of the same, which could be—and were—run on the Polymath system:

Citation	Publication Date	Bates Number
Eric Fijalkow, <i>A Numerical Solution To The Vlasov Equation</i> , 116 COMPUTER PHYSICS COMMUNICATIONS 319-328 (1999)	1999	NVIDIA-ACS-1379970
M. Bassi, U. Becciani, U. Lombardo, G. Russo, N. Sandulescu, W. Zuo, <i>A parallel code for the kinetic Landau-Vlasov transport equation</i> , 118 COMPUTER PHYSICS COMMUNICATIONS 110-118 (1999)	1999	NVIDIA-ACS-1379980
Stefan Goedecker, Mireille Boulet, Thierry Deutsch, <i>An efficient 3-dim FFT for plane wave electronic structure calculations on massively parallel machines composed of multiprocessor nodes</i> , 154 COMPUTER PHYSICS COMMUNICATIONS 105-110 (2003)	2003	NVIDIA-ACS-1380013
C. Othmer, J. Schüle, <i>Dynamic load balancing of plasma particle-in-cell simulations: The taskfarm alternative</i> , 147 COMPUTER PHYSICS COMMUNICATIONS 741-744 (2002)	2002	NVIDIA-ACS-1380031
Quan Ming Lu, Dong Sheng Cai, <i>Implementation of parallel plasma particle-in-cell codes on PC cluster</i> , 135 COMPUTER PHYSICS COMMUNICATIONS 93-104 (2001)	2001	NVIDIA-ACS-1380035
Stephen Bergeron, Alain Vincent, <i>Implementation strategies for real-time particle transport solver</i> , 120 COMPUTER PHYSICS COMMUNICATIONS 177-184 (1999)	1999	NVIDIA-ACS-1380047
Edward S. Smyth 1, Jonathan S. Parker, K.T. Taylor, <i>Numerical integration of the time-dependent Schrodinger equation for laser-driven helium</i> , 114 COMPUTER PHYSICS COMMUNICATIONS 1-14 (1998)	1998	NVIDIA-ACS-1380055
Eric Fijalkow, <i>Numerical solution to the Vlasov equation: 1D Code</i> , 116 COMPUTER PHYSICS COMMUNICATIONS 329-335 (1999)	1999	NVIDIA-ACS-1380069

Citation	Publication Date	Bates Number
Eric Fijalkow, <i>Numerical solution to the Vlasov equation: 2D Code</i> , 116 COMPUTER PHYSICS COMMUNICATIONS 336-344 (1999)	1999	NVIDIA-ACS-1380076
Olivier Coulaud, Eric Sonnendrücker, Eric Dillon, Pierre Bertrand, Alain Ghizzo, <i>Parallelization of Semi-Lagrangian Vlasov Codes</i> , JOURNAL OF PLASMA PHYSICS (1999)	1999	NVIDIA-ACS-1380085
Vlasov Textbooks	Pre-2005	POLYMATH_0009658
Matlab - CDMA Reference Blockset (2004)	2004	NVIDIA-ACS-1340048
Matlab - Communications Toolbox User Guide (2002)	2002	NVIDIA-ACS-1350269
Matlab - Control System Toolbox User Guide (2005)	2005	NVIDIA-ACS-1340204
Matlab - Data Acquisition Toolbox Quick Reference Guide (2004)	2004	NVIDIA-ACS-1340552
Matlab - Data Acquisition Toolbox User Guide (2004)	2004	NVIDIA-ACS-1340566
Matlab - Dial and Gauges Blockset User Guide (2004)	2004	NVIDIA-ACS-1341144
Matlab - External Interface-API V6 (2001)	2001	NVIDIA-ACS-1350678
Matlab - Getting Started with Control System Toolbox (2002)	2002	NVIDIA-ACS-1351151
Matlab - Getting Started with MATLAB (2002)	2002	NVIDIA-ACS-1351329
Matlab - MAT-File Format User Guide (1999)	1999	NVIDIA-ACS-1351467
Matlab - MAT-File Format User Guide (2004)	2004	NVIDIA-ACS-1341284
Matlab - Model Control Toolbox (1998)	1998	NVIDIA-ACS-1351894
Matlab - Neural Network Toolbox (2003)	2003	NVIDIA-ACS-1341329
Matlab - Nonlinear Control Design Blockset User Guide (2002)	2002	NVIDIA-ACS-1352144
Matlab - Optimization Toolbox (2003)	2003	NVIDIA-ACS-1342169
Matlab - Partial Differential Equation Toolbox User Guide (2002)	2002	NVIDIA-ACS-1352252
Matlab - Real-Time Windows Target User Guide (2002)	2002	NVIDIA-ACS-1342521
Matlab - Report Generator User Guide (2004)	2004	NVIDIA-ACS-1342699
Matlab - Requirements Management Interface (2000)	2000	NVIDIA-ACS-1343019
Matlab - Robust Control Toolbox User Guide (2004)	2004	NVIDIA-ACS-1343097

Citation	Publication Date	Bates Number
Matlab - Signal Processing Blockset Users Guide (2004)	2004	NVIDIA-ACS-1343329
Matlab - Signal Processing Toolbox User Guide (2004)	2004	NVIDIA-ACS-1344792
Matlab - Spline Toolbox (2003)	2003	NVIDIA-ACS-1345799
Matlab - Symbolic Math Toolbox (2005)	2005	NVIDIA-ACS-1346019
Matlab - System Identification Toolbox (2004)	2004	NVIDIA-ACS-1346310
Matlab - Using Matlab (2001)	2001	NVIDIA-ACS-1352542
Matlab - Using MATLAB (2002)	2002	NVIDIA-ACS-1346717
Matlab - Using MATLAB Graphics V7 (2005)	2005	NVIDIA-ACS-1347897
Matlab - Wavelet Toolbox (2003)	2003	NVIDIA-ACS-1353446
Matlab - xPC Target I-O Reference Guide (2004)	2004	NVIDIA-ACS-1348604
Matlab - xPC Target User Guide (2003)	2003	NVIDIA-ACS-1350070
<i>A Distributed System Architecture For A Distributed Application Environment</i> (1994)	1994	NVIDIA-ACS-1379943
<i>A Self-Organizing Flock of Condors</i> (2003)	2003	NVIDIA-ACS-1379989
<i>A worldwide flock of Condors Load sharing among workstation clusters</i> (1996)	1996	NVIDIA-ACS-1380000
<i>Distributed Simulation of Parallel DSP Architectures on Workstation Clusters</i> (1996)	1996	NVIDIA-ACS-1380019
Matlab - Matlab Compiler (1999)	1999	NVIDIA-ACS-1351508
Chu <i>et al.</i> , <i>FFT algorithms and their adaptation to parallel processing</i> , LINEAR ALGEBRA AND ITS APPLICATIONS (1998)	1998	NVIDIA-ACS-0163554
Agarwal, <i>et al.</i> , "A High Performance Parallel Algorithm for 1-D FFT," Supercomputing '94 (1994)	1994	NVIDIA-ACS-0163536
Frigo and Johnson, <i>FFTW User's Manual</i> (March 16, 2003)	March 16, 2003	NVIDIA-ACS-0163860
Frigo and Johnson, <i>The Fastest Fourier Transform in the West</i>	Sept. 11, 1997	NVIDIA-ACS-0165490
Cramer, Board, <i>The Development and Integration of a Distributed 3D FFT for a Cluster of Workstations</i> , Proceedings of the 4 th Annual Linux Showcase and Conference, USENIX, 2000	2000	NVIDIA-ACS-0163589
U.S. Patent Application Pub. No. US 2003/0195938 A1, Kevin David Howard	2003	NVIDIA-ACS-0165342
U.S. Patent No. 7,162,590	Jan. 6, 2005	NVIDIA-ACS-0170403

Citation	Publication Date	Bates Number
Anshuman Nayak, <i>et al.</i> , <i>A Library based compiler to execute MATLAB Programs on a Heterogeneous Platform</i> , Symp. on FPGA Custom Computing Machines (FCCM-2000) Napa Valley, CA, Apr. 2000	2000	NVIDIA-ACS-0165016
U.S. Patent No. 7,694,158	Nov. 16, 2006	NVIDIA-ACS-0170423
Fan <i>et al.</i> , <i>GPU Cluster for High Performance Computing</i> , SC'04, November 6-12, 2004	Nov. 6, 2004	NVIDIA-ACS-0172335

SPECIAL PURPOSE PROCESSORS AND PROCESSORS CONTAINING MULTIPLE CORES

The following exemplary references describe processors containing multiple cores and/or special purpose processors, which were conventionally known in the art to increase performance, thereby motivating a POSITA to combine with Polymath without undue experimentation:

Citation	Publication Date	Bates Number
Chu <i>et al.</i> , <i>FFT algorithms and their adaptation to parallel processing</i> , LINEAR ALGEBRA AND ITS APPLICATIONS (1998)	1998	NVIDIA-ACS-0163554
Agarwal, <i>et al.</i> , <i>A High Performance Parallel Algorithm for 1-D FFT</i> , SUPERCOMPUTING '94 (1994)	1994	NVIDIA-ACS-0163536
Frigo and Johnson, <i>FFTW User's Manual</i> (March 16, 2003)	March 16, 2003	NVIDIA-ACS-0163860
Frigo and Johnson, <i>The Fastest Fourier Transform in the West</i>	Sept. 11, 1997	NVIDIA-ACS-0165490
Cramer, Board, <i>The Development and Integration of a Distributed 3D FFT for a Cluster of Workstations</i> , Proceedings of the 4 th Annual Linux Showcase and Conference, USENIX, 2000	2000	NVIDIA-ACS-0163589
U.S. Patent Application Pub. No. US 2003/0195938 A1, Kevin David Howard	2003	NVIDIA-ACS-0165342
U.S. Patent No. 7,162,590	Jan. 6, 2005	NVIDIA-ACS-0170403
Anshuman Nayak, <i>et al.</i> , <i>A Library based compiler to execute MATLAB Programs on a Heterogeneous Platform</i> , Symp. on FPGA Custom Computing Machines (FCCM-2000) Napa Valley, CA, Apr. 2000	2000	NVIDIA-ACS-0165016
U.S. Patent No. 7,694,158	Nov. 16, 2006	NVIDIA-ACS-0170423
Fan <i>et al.</i> , <i>GPU Cluster for High Performance Computing</i> , SC'04, November 6-12, 2004	Nov. 6, 2004	NVIDIA-ACS-0172335



ADDITIONAL MOTIVATION TO COMBINE

A POSITA would have been motivated to combine the Polymath system using the AppleSeed configuration and Pooch with other software and applications (as well as the Launch Den Mother, Launch Puppy, and SEM references) together. The teaching of the Polymath system using the AppleSeed configuration and Pooch and the features described therein were actually combined into a single project and system provides a POSITA with a strong, express motivation to combine those features from related publications. Moreover, multiple references share Dr. Viktor Decyk and Dean Dauger as authors, and were publicly available through <http://daugerresearch.com> and <http://exodus.physics.ucla.edu>.

A POSITA would have been further motivated to combine the Polymath system using the AppleSeed configuration and Pooch with computers (including Apple Mac and Xserve computers) containing multi-core processors and co-processors. It would have been obvious and a natural step to implement the Polymath system using a hardware processor containing a plurality of processing cores for one or more of the nodes. A POSITA would have been motivated to do so at least because multi-core processors can be faster and save power compared to multi-processor computers. Dr. Dauger confirmed in contemporaneous statements that Polymath system using the AppleSeed configuration and Pooch would benefit from incremental improvements to Apple Mac and other computer systems used by Polymath. *See, e.g.*, NVIDIA-ACS-1137689 (Dauger Research Profiled on National Television Part 1); NVIDIA-ACS-1137688 (Dauger Research Profiled on National Television Part 2), and NVIDIA-ACS-1137687 (Dauger Research Profiled on National Television Part 3); NVIDIA-ACS-1137686 (Apple WWDC 2004 Session 642). Polymath using the AppleSeed configuration and Pooch references themselves describe the abilities of Multiprocessor Macintosh and provided MacMP API for multi-core processors. In addition, during this timeframe, multi-core processors were prevalent in the marketplace, and it would have been a matter of mere design choice to use computers that contained hardware processors containing multiple processing cores. Thus, computers containing hardware processors with multiple processing cores were available during the relevant timeframe, and a POSITA would have known how to create a cluster where some or all of the nodes comprised a hardware processor with multiple cores. A POSITA would have been motivated to do this because multi-core processors were known to be faster than single-core processors for many tasks. A POSITA also would have been motivated to do so as a matter of mere design choice and because hardware processors with multiple cores were readily available in the marketplace.

Asserted Claim	Prior Art Disclosure
	<p>NVIDIA-ACS-1320503 (Inside Macintosh – Networking), at _053; <i>id.</i> at _054 (“The PPCInform, PPCRead, and PPCWrite functions should always be executed asynchronously, because they require interaction from the other application in the session before they complete execution.”)</p> <div data-bbox="619 358 1766 686" style="border: 1px solid black; padding: 10px; margin: 10px 0;"> <p>The PPCStart function initiates a session with the destination port specified in the name and location fields.</p> <pre>FUNCTION PPCStart (pb: PPCStartPBPtr; async: Boolean): OSErr;</pre> <p>pb A pointer to a PPCStart parameter block.</p> <p>async A value that specifies whether the function is to be executed asynchronously (TRUE) or synchronously (FALSE).</p> </div> <p>NVIDIA-ACS-1320503 (Inside Macintosh – Networking), at _099.</p> <p><i>See also generally</i> NVIDIA-ACS-1323962 (Inside Macintosh Vol. VI) (discussing various PPC Toolbox and AppleTalk communications protocols and functions that can and should be run asynchronously).</p> <p>Both the Polymath 2000 Cluster and the Polymath 2002 Cluster ran Pooch, including MacMPI and AppleTalk. <i>See, e.g.</i>, Claim 1.Pre; <i>see also</i> POLYMATH_0076119; POLYMATH_0076120; POLYMATH_0076121; POLYMATH_0003561.</p> <p>Polymath is “a numerically-intensive parallel computing cluster using Power Macintosh hardware and the Macintosh Operating System” ACS_NVIDIA_005621, at _5621. The system was described as using “shrink-wrap applications, such as IDL and Mathematica, to visualize the output of our code.” ACS_NVIDIA_005621, at _5628.</p> <p>By 2005, the Pooch software component was described as “a parallel computing and cluster management tool that can organize a job’s files into subdirectories on other nodes and retrieve files on those nodes containing output from completed jobs” in “a wide variety of parallel programming environments ...” ACS_NVIDIA_005607, at _5609. Appleseed implemented as “Mac clusters don’t have a head node, at least not in the sense of the head node found in typical Linux-based clusters. There is no permanent controlling unit or units, so the behavior is much more peer to peer.” ACS_NVIDIA_005607, at _5611.</p> <p><i>See also</i> built in mathematica functions run on the Polymath 2000 and Polymath 2002 that allow for a “peer-to-peer” connection: POLYMATH_0080235 (addtwo.c); POLYMATH_0080237 (bitops.c); POLYMATH_0080239 (counter.tm); POLYMATH_0080248 (reverse.tm); POLYMATH_0080249 (sumalist.tm).</p> <p>At WWDC 2004, Dean Dager explained that Pooch and MacMPI used “the all-to-all communication pattern where every node is communicating to every other one.” The all-to-all communication pattern is “very important for data transposes of matrices, and</p>

Asserted Claim	Prior Art Disclosure
	<p>that's important, say, for performing a 1-D, a very large 1-D FFT in parallel. You have to go through data transposes and consequently the message passing patterns. You have a lot of all the all communicate.”</p> <div data-bbox="661 321 1717 1052" data-label="Diagram"> <p>The diagram, titled "Message-Passing Patterns Supported via MPI", illustrates various communication topologies. It includes: <ul style="list-style-type: none"> Master-Slave: A central node connected to multiple peripheral nodes. All to All: A fully connected mesh of nodes. Tree: A hierarchical structure with a root node and branching children. Nearest Neighbor: A linear chain of nodes connected sequentially. Irregular: A non-uniform, complex network of connections. Or Combinations: A complex hybrid structure combining elements of the other patterns. </p> <p>NVIDIA-ACS-1137686, at 47:21–47:52; <i>see also id.</i> (“Bold on the lower right, the quantum pick simulate some diagram on a PIC simulation is in this case. This is showing a two-dimensional quantum wave function in a simple harmonic oscillator and showing the circulation of the electron around the wave function. This is actually work that was based on my doctoral dissertation, which I did entirely on Mac clusters and what it involves is an approximation of Feynman path integrals that to be able to choose sample, just the all the possible classical paths and use a plasma code to be able to push those paths forward and determine the evolution of a quantum wavefunction.).</p> <p><i>See also</i> “FIG. 9 depicts an embodiment of a 3-node cluster computer ... with the first computing device 30A, we can see that it comprises a processor 12A and a user interface 16 ... [a] second computing device 30B ... [and a] third computing devices 30C” NVIDIA-ACS-0166349 (’931 Pub.), 0048-0050. “[I]t is known that other parallel applications compute in a peer-to-peer arrangement as well as between the Node n’s and Node 0. The CNCI is capable of operating with both example parallel application forms.” NVIDIA-ACS-0166349 (’931 Pub.), 0045.</p> </div>

EXHIBIT S

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

ADIDAS AG,

Plaintiff,

v.

UNDER ARMOUR, INC. and
MAPMYFITNESS, INC.,

Defendants.

C.A. No. 14-130-GMS

ORDER

WHEREAS, the plaintiff, adidas AG (“adidas”), filed a Complaint (D.I. 1) on February 4, 2014, a First Amended Complaint (D.I. 10) on March 14, 2014, and a Second Amended Complaint (D.I. 44) on September 11, 2014 against the defendants, Under Armour, Inc. (“Under Armour”) and MapMyFitness, Inc. (“MapMyFitness”) alleging that the defendants infringed U.S. Patent Nos. 7,292,867 (“the ’867 Patent”); 7,805,149 (“the ’149 Patent”); 7,941,160 (“the ’160 Patent”); 7,957,752 (“the ’752 Patent”); 8,068,858 (“the ’858 Patent”); 8,244,226 (“the ’226 Patent”); 7,905,815 (“the ’815 Patent”); 7,931,562 (“the ’562 Patent”); 8,092,345 (“the ’345 Patent”); 8,579,767 (“the ’767 Patent”); 8,725,276 (“the ’276 Patent”); 8,721,502 (“the ’502 Patent”); and 8,652,009 (“the ’009 Patent”);

WHEREAS, Under Armour and MapMyFitness filed Answers and Counterclaims to the First Amended Complaint on March 31, 2014 (D.I. 13; D.I. 15) and to the Second Amended Complaint on June 30, 2015 (D.I. 170; D.I. 171), denying infringement of the asserted patents and seeking declaratory judgments of non-infringement, invalidity, and unenforceability;

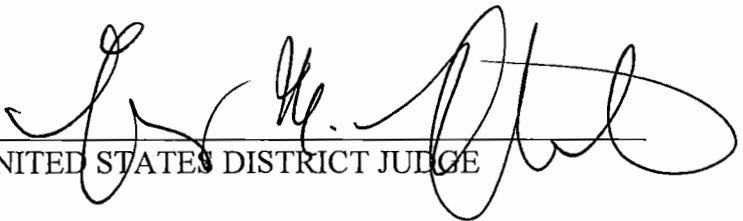
WHEREAS, the court held a Markman Hearing on May 14, 2015, and issued its claim construction order construing the terms on June 15, 2015 (D.I. 161);

WHEREAS, presently before the court is the defendants' Motion to Modify the Court's Claim Construction Order (D.I. 189), the plaintiff's response thereto (D.I. 196), and the defendants' reply (D.I. 199);

WHEREAS, having considered the party's positions as set forth in their papers, the pleadings, as well as the applicable law;

IT IS HEREBY ORDERED that the defendants' Motion to Modify the Court's Claim Construction Order (D.I. 189) is DENIED.¹

Dated: December 15, 2015


UNITED STATES DISTRICT JUDGE

¹ The defendants request the court to modify its construction of the term "route path" as claimed in the '858 Patent. Their argument centers on actions taken by the Patent Trial and Appeal Board ("PTAB"). On February 5, 2015, Under Armour filed a Petition for *inter partes* review ("IPR") of the '858 Patent. (D.I. 190, Ex. 1.) In its Petition, Under Armour construed "with respect to a route path" to mean "in relation to a predetermined path" and argued that based on that construction, the challenged claims were anticipated by the prior art. (*Id.* at 10–11.) Under Armour's construction was consistent with the construction rejected by the court in its claim construction ruling. (D.I. 161.) Adidas filed a Preliminary Response to the Petition on May 20. (D.I. 190, Ex. 2.) In its response, Adidas stated that it disagreed with Under Armour's construction of "with respect to a route path," but did not propose any construction of its own. Adidas then argued that under Under Armour's construction, the patented claims were not anticipated. The PTAB agreed with Adidas and denied the Petition on August 3, 2015. (D.I. 190, Ex. 3.) The defendants argue that because the PTAB relied on a different claim construction than the court, collateral estoppel should apply.

The PTAB's choice not to institute an IPR is not the type of adjudication that leads to issue preclusion. In declining to institute the IPR, the PTAB did not reach a final decision on the construction of "with respect to a route path." Without analysis, the PTAB adopted Under Armour's construction "for purposes of this decision." (D.I. 190, Ex. 3 at 9.) The PTAB found that using Under Armour's own construction, it had not shown a reasonable likelihood of success in its invalidity contentions. (*Id.* at 9–14.) Furthermore, the patent owner did not present the PTAB with a different construction than the one it presented to the court. The fact that it did not provide the PTAB with *any* construction does not weigh in favor of issue preclusion in such a preliminary proceeding. Contrary to the defendants' assertions, the patentee is not "trying to have its cake and eat too." (D.I. 190 at 11). The court is not bound by a preliminary claim construction used by the PTAB for the limited purpose of denying an IPR request.

With no equitable basis for collateral estoppel, the defendants' motion is essentially a motion for reconsideration. The purpose of a motion for reconsideration is to "correct manifest errors of law or fact or to present newly discovered evidence." *Max's Seafood Café ex rel. Lou-Ann, Inc. v. Ouinteros*, 176 F.3d 669, 677 (3d Cir. 1999). Accordingly, a court may alter or amend its judgment if the movant demonstrates at least one of the following: (1) a change in the controlling law; (2) availability of new evidence not available when the decision issued; or (3) a need to correct a clear error of law or fact or to prevent manifest injustice. *Id.* The court finds no basis to modify its original claim construction ruling, and thus declines to do so.

EXHIBIT T

IPR2021-00019

Petition for *Inter Partes* Review

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

NVIDIA CORPORATION

Petitioner

v.

ADVANCED CLUSTER SYSTEMS, INC.

Patent Owner

Case IPR2021-00019

Patent No. 10,333,768

PETITION FOR INTER PARTES REVIEW OF

U.S. PATENT NO. 10,333,768

UNDER 35 U.S.C. § 312 AND 37 C.F.R. § 42.104

TABLE OF CONTENTS

I.	Introduction.....	1
II.	Notices, Statements, and Payment of Fees	3
	A. Real Party In Interest Under 37 C.F.R. § 42.8(b)(1).....	3
	B. Related Matters Under 37 C.F.R. § 42.8(b)(2)	3
	C. Lead and Back-Up Counsel Under 37 C.F.R. § 42.8(b)(3)	3
	D. Service Information Under 37 C.F.R. § 42.8(b)(4).....	4
III.	Grounds For Standing.....	4
IV.	Fees Under 37 C.F.R. § 42.103	4
V.	The '768 Patent and Prosecution History	4
	A. Summary of the '768 Patent.....	4
	B. The '768 Prosecution History	7
VI.	Identification of Challenges Under 37 C.F.R. § 42.104(B)	7
	A. Prior Art.....	7
	B. Challenges	10
VII.	This Petition Is Proper Under 35 U.S.C. §§ 314(a) AND 325(d)	11
	A. Stay	11
	B. Trial Date.....	11
	C. Parallel Proceeding.....	12
	D. Issue Overlap.....	12
	E. Same Party.....	13
	F. Other Considerations.....	13
VIII.	Level of Ordinary Skill in the Art	14

IX.	Claim Construction.....	14
X.	Detailed Explanation and Supporting Evidence.....	14
A.	Ground 1: Schreiner1 in View of Schreiner2, Schreiner3, the Distributed Maple Code, the Maple Guide, the SPARC IV Article, and the AMD Article Renders Obvious Claims 1, 4-10, 18-22, 24-25, 30-31, and 33-34.....	14
1.	Motivations to Combine	15
2.	Overview of the Distributed Maple Publications	17
3.	Claim 1:.....	23
a.	“A computer cluster comprising:”	23
b.	“a plurality of nodes, wherein each of the plurality of nodes comprises a hardware processor,”	23
c.	“wherein one or more of the nodes are configured to receive a command to start a cluster initialization process for the computer cluster, and”	25
d.	“wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, is capable of causing the hardware processor to evaluate mathematical expressions; and”	26
e.	“a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture;”	28
f.	“wherein the plurality of nodes comprises: a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and”	32
i.	“a first node comprising a first hardware processor”	33

- ii. “configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel” 33
 - iii. “the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution” 34
 - g. “a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of the first mathematical expression evaluation to a third node;” 38
 - i. “plurality of processing cores” 38
 - ii. “the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of the first mathematical expression evaluation to a third node” 41
 - h. “wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;” 43
 - i. “plurality of processing cores” 44
 - ii. “the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node” 44

- i. “wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface;” 48
- j. “wherein one or more of the nodes are configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to one or more single-node kernels.” 49
 - i. “accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other” 49
 - ii. “after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to one or more single-node kernels.” 50
- 4. Claim 4: “The computer cluster of claim 1, wherein each of the nodes comprises one or more cluster node modules.”52
- 5. Claim 5: “The computer cluster of claim 1, wherein each single-node kernel is stored in a non-transitory computer-readable medium and configured to accept and execute a request.”55
- 6. Claim 6: “The computer cluster of claim 5, wherein each of the nodes comprises one or more cluster node modules, wherein each of the cluster node modules comprises instructions stored in a non-transitory computer-readable medium, and wherein the instructions, when executed by the hardware processor, cause the cluster node module to communicate with the single-node kernel and with one or more other cluster node modules.”56

7. Claim 7: “The computer cluster of claim 6, wherein the plurality of cluster node modules act as a cluster.”57
8. Claim 8: “The computer cluster of claim 6, wherein the plurality of cluster node modules communicate with one another to act as a cluster.”57
9. Claim 9: “The computer cluster of claim 8, wherein the computer cluster includes the user interface.”57
10. Claim 10: “The computer cluster of claim 9, wherein each cluster node module accepts instructions from the user interface and interprets one or more of the instructions.”58
11. Claim 18: “The computer cluster of claim 1, wherein one or more of the nodes are configured to communicate at least some of the user instructions.”60
12. Claim 19: “The computer cluster of claim 18, wherein one or more of the nodes are configured to communicate at least some of the user instructions to one or more single-node kernels.”60
13. Claim 20: “The computer cluster of claim 1, wherein one or more of the nodes are configured to accept user instructions via one or more of the nodes.”60
14. Claim 21: “The computer cluster of claim 20, wherein one or more of the nodes are configured to communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other.”61
15. Claim 22: “The computer cluster of claim 1, wherein one or more of the nodes are configured to transmit at least some of the user instructions that originate from a user interface.”61

16. Claim 24: “The computer cluster of claim 1, wherein one or more of the nodes are configured to accept user instructions before communicating at least some of the user instructions to one or more single-node kernels.”.....62
17. Claim 25: “The computer cluster of claim 1, wherein the plurality of nodes are configured to communicate with one another to interpret and translate commands for execution by a plurality of single-node kernels.”62
18. Claim 30: “The computer cluster of claim 4, wherein each of the plurality of nodes implements asynchronous calls that enable the single-node kernel to perform computation tasks while the cluster node modules are simultaneously communicating with one another.”.....64
19. Claim 31: “The computer cluster of claim 4, wherein intercommunication among the plurality of single-node kernels during thread execution is enabled by the plurality of cluster node modules, and wherein the computer cluster is configured to permit exchange of information between nodes during the course of a parallel computation.”68
20. Claim 33: “The computer cluster of claim 1, wherein the plurality of nodes are configured to permit exchange of information between nodes during the course of parallel computation.”70
21. Claim 34: “The computer cluster of claim 1, wherein each of the plurality of nodes comprises instructions executable by the hardware processor and configured to implement asynchronous behavior, wherein the instructions comprise: a first instruction to asynchronously send a payload to another node; a second instruction to asynchronously receive a payload from another node; and a third instruction to search for a payload matching a message specifier.”71

1.	Claim 30: “The computer cluster of claim 4, wherein each of the plurality of nodes implements asynchronous calls that enable the single-node kernel to perform computation tasks while the cluster node modules are simultaneously communicating with one another.”.....	75
2.	Claim 31: “The computer cluster of claim 4, wherein intercommunication among the plurality of single-node kernels during thread execution is enabled by the plurality of cluster node modules, and wherein the computer cluster is configured to permit exchange of information between nodes during the course of a parallel computation.”	77
3.	Claim 34: “The computer cluster of claim 1, wherein each of the plurality of nodes comprises instructions executable by the hardware processor and configured to implement asynchronous behavior, wherein the instructions comprise: a first instruction to asynchronously send a payload to another node; a second instruction to asynchronously receive a payload from another node; and a third instruction to search for a payload matching a message specifier.”	77
XI.	Conclusion	78

VIII. LEVEL OF ORDINARY SKILL IN THE ART

A POSITA as of the filing date of the '768 Patent would have had a Bachelor's degree in computer science, electrical engineering, or an equivalent field, and two years of academic or industry experience working with distributed computing. EX-1005, ¶¶38-40.

IX. CLAIM CONSTRUCTION

The Challenged Claims are interpreted using the same standard used in federal district court. 37 C.F.R. § 42.100(b). Petitioner believes that no express construction of any term is needed to resolve the challenges herein but reserves the right to present express constructions in response to any argument by the Patent Owner. Petitioner also reserves the right to present constructions at a later time, including district court litigation, that differ, in whole or in part, from any constructions presented during these proceedings.

X. DETAILED EXPLANATION AND SUPPORTING EVIDENCE

A. Ground 1: Schreiner1 in View of Schreiner2, Schreiner3, the Distributed Maple Code, the Maple Guide, the SPARC IV Article, and the AMD Article Renders Obvious Claims 1, 4-10, 18-22, 24-25, 30-31, and 33-34.

Claims 1, 4, 7-10, 18-22, 24-25, 30-31, and 33-34 are rendered obvious by Schreiner1 in view of Schreiner2, Schreiner3, the Distributed Maple Code, the Maple Guide, the SPARC IV Article, and the AMD Article. EX-1005, ¶43.

IPR2021-00019

Petition for *Inter Partes* Review

implement the scheduler messaging protocol (the first and second instructions) and the result table search and match function (the third instruction). EX-1005, ¶162.

XI. CONCLUSION

For these reasons, Petitioner requests *inter partes* review of the '768 patent's claims 1, 4-10, 18-22, 24-25, 30-31, and 33-34.

By: /Brent Yamashita/
Brent Yamashita

EXHIBIT U

IPR2021-00020

Petition for *Inter Partes* Review

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

NVIDIA CORPORATION

Petitioner

v.

ADVANCED CLUSTER SYSTEMS, INC.

Patent Owner

Case IPR2021-00020

Patent No. 10,333,768

**PETITION FOR INTER PARTES REVIEW OF
U.S. PATENT NO. 10,333,768
UNDER 35 U.S.C. § 312 AND 37 C.F.R. § 42.104**

TABLE OF CONTENTS

I.	Introduction.....	1
II.	Mandatory Notices Under 37 C.F.R. § 42.8(b)(1)-(4)	3
	A. Real Party In Interest.....	3
	B. Related Matters.....	3
	C. Counsel	3
	D. Service Information	4
III.	Standing	4
IV.	Fees	4
V.	The '768 Patent and Prosecution History	4
	A. Summary of the '768 Patent.....	4
	B. The '768 Prosecution History	7
VI.	Identification of Challenges Under 37 C.F.R. § 42.104(B)	7
	A. Prior Art.....	7
	B. Challenges	12
VII.	This Petition is Proper Under 35 U.S.C. §§ 314(a) AND 325(d)	13
	A. Stay	13
	B. Trial Date.....	14
	C. Parallel Proceeding.....	14
	D. Issue Overlap	15
	E. Same Party	15
	F. Other Considerations.....	15

VIII.	Level of Ordinary Skill in the Art	16
IX.	Claim Construction.....	16
X.	Detailed Explanation and Supporting Evidence.....	17
A.	Ground 1: Schreiner ¹ in View of Schreiner ² , Schreiner ³ , the Distributed Maple Code, the Maple Guide, the SPARC IV Article, and the AMD Article Renders Obvious Claims 26-27, 29, 35, and 37.	17
1.	Motivations to Combine	17
2.	Overview of the Distributed Maple Publications	19
3.	Claim 26:.....	24
a.	“A computer cluster comprising:”	24
b.	“a plurality of nodes, wherein one or more of the nodes are configured to receive: a command to start a cluster initialization process for the computer cluster, wherein the cluster initialization process comprises establishing communication among two or more of the nodes; and an instruction from a user interface or a script; and”	25
i.	“a command to start a cluster initialization process for the computer cluster, wherein the cluster initialization process comprises establishing communication among two or more of the nodes”	26
ii.	“an instruction from a user interface or a script”	26
c.	“a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using asynchronous calls;”	28
d.	“wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel	

- that, when executed, is capable of causing a hardware processor to evaluate mathematical expressions;” 34
- e. “wherein the plurality of nodes comprises: a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and..... 36
- i. configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, 37
- ii. the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and 38
- f. “a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of mathematical expression evaluation to a third node;” 41
- i. “a plurality of processing cores” 42
- ii. “the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of mathematical expression evaluation to a third node” 44
- g. “wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to

- receive the result of mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;” 46
- i. “a plurality of processing cores” 47
- ii. the third node is configured to receive the result of mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;” 47
- h. “wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface or the script;” 50
- i. “wherein one or more of the nodes are configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to one or more single-node kernels.” 51
- i. “accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other” 51
- ii. “after communicating at least some of the user instructions using the mechanism, communicate at least some of the user

	instructions to one or more single-node kernels.”	52
4.	Claim 27: “The computer cluster of claim 26, wherein the asynchronous calls comprise a first command to create a first packet containing: an expression to be sent as payload; and a target node where the expression should be sent; wherein the first command is configured to be called from within a single-node kernel; wherein the single-node kernel is configured to send the first packet to a local cluster node module; and wherein the local cluster node module is configured to forward the expression to the target node.”	53
5.	Claim 29	56
6.	Claim 35	56
a.	“A computer cluster node for evaluating expressions in parallel with other computer cluster nodes, the computer cluster node comprising:”	56
b.	“a hardware processor configured to access one or more non-transitory memory devices comprising program code for a single-node kernel that, when executed, causes the hardware processor to interpret user instructions, to evaluate mathematical expressions, and to produce results of mathematical expression evaluation, wherein the hardware processor comprises multiple processor cores;”	57
c.	“a user connection interface configured to receive a command to start a cluster initialization process for a computer cluster;”	58
d.	“a mechanism to communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture; and”	60
e.	“program code that, when executed, is capable of causing the hardware processor to: receive calls	

from a second node comprising a second hardware processor configured to access a second memory comprising program code for a user interface and program code for a second single-node kernel, the second single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; execute, using the hardware processor, at least a first mathematical expression evaluation; and communicate a result of the first mathematical expression evaluation to a third node comprising a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the computer cluster node, execute at least a second mathematical expression evaluation using the result of the first mathematical expression evaluation, and communicate a result of the second mathematical expression evaluation to the first node;” 60

- i. receive calls from a second node comprising a second hardware processor configured to access a second memory comprising program code for a user interface and program code for a second single-node kernel, the second single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution;..... 62
- ii. execute, using the hardware processor, at least a first mathematical expression evaluation; and 62
- iii. communicate a result of the first mathematical expression evaluation to a third node comprising a third hardware processor with a plurality of processing cores,..... 63

- iv. wherein the third node is configured to receive the result of mathematical expression evaluation from the computer cluster node, execute at least a second mathematical expression evaluation using the result of the first mathematical expression evaluation, and communicate a result of the second mathematical expression evaluation to the first node;” 63
- f. “wherein the user connection interface is configured to return at least one result of mathematical expression evaluation to a user interface or a script; and” 64
- g. “wherein the computer cluster node is configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to the single-node kernel.” 64
- 7. Claim 37: “The computer cluster node of claim 35, wherein the computer cluster node is configured to permit exchange of information with other computer cluster nodes during the course of parallel computation.”65
- B. Ground 2: The References in Ground 1 in Further View of the MPI Standard Render Obvious Claims 26-27 and 37.....65
 - 1. Claim 26 (Excerpted Above):66
 - 2. Claim 27 (Excerpted Above)67
 - 3. Claim 37 (Excerpted Above)67
- C. Ground 3: The References in Ground 1 in Further View of the Maple Reference and Howard Render Obvious Claims 36 and 39.....68

1.	Claim 36: “The computer cluster node of claim 35, wherein the one or more non-transitory memory devices comprise program code for performing a parallel fast Fourier transform, wherein the program code causes the hardware processor to evaluate a command to perform a Fourier transform on an array comprising a first data portion that is stored on the computer cluster node and a second data portion that is not stored on the computer cluster node.”	68
2.	Claim 39: “The computer cluster node of claim 35, wherein the hardware processor comprises a special purpose microprocessor.”	73
D.	Ground 4: The References in Ground 3 in Further View of the FFTW Manual Render Obvious Claim 36.....	74
1.	Claim 36 (Excerpted Above)	74
E.	Ground 5: The References in Ground 1 in Further View of PC Magazine 1 and PC Magazine 2 Render Obvious Claim 39.	75
1.	Claim 39 (Excerpted Above)	75
F.	Ground 6: The References in Ground 2 in Further View of Nayak Render Obvious Claim 39.....	77
1.	Claim 39 (Excerpted Above)	77
XI.	Conclusion	78

has never been challenged in a post-issuance proceeding, all but one of the references were not cited during prosecution of the '768 patent, and the teachings used here from the only cited reference were not contested during the original prosecution.

In sum, all but one of the *Fintiv* factors weigh strongly against a discretionary denial and the remaining factor is the “same party” factor that applies to nearly all petitions and should be given little weight. Accordingly, the Board should not issue a discretionary denial.

VIII. LEVEL OF ORDINARY SKILL IN THE ART

A POSITA as of the filing date of the '768 Patent would have had a Bachelor's degree in computer science, electrical engineering, or an equivalent field, and two years of academic or industry experience working with distributed computing. EX-1105, ¶¶38-40.

IX. CLAIM CONSTRUCTION

The Challenged Claims are interpreted using the same standard used in federal district court. Petitioner believes that no express construction of any term is needed to resolve the challenges herein but reserves the right to present express constructions in response to any argument by the Patent Owner. Petitioner also reserves the right to present constructions at a later time, including district court

litigation, that differ, in whole or in part, from any constructions presented during these proceedings.

X. DETAILED EXPLANATION AND SUPPORTING EVIDENCE

A. Ground 1: Schreiner1 in View of Schreiner2, Schreiner3, the Distributed Maple Code, the Maple Guide, the SPARC IV Article, and the AMD Article Renders Obvious Claims 26-27, 29, 35, and 37.

All elements of claims 26-27, 29, 35, and 37 are disclosed in and rendered obvious by Schreiner1 in view of Schreiner2, Schreiner3, the Distributed Maple Code, the Maple Guide, the SPARC IV Article, and the AMD Article.

1. Motivations to Combine

This Ground is based on publications related to the same project, called “Distributed Maple,” led by Professor Schreiner at Johannes Kepler University in Linz, Austria. Schreiner began the project in 1998, developing the Distributed Maple system “as a work platform for parallel and networked environments.” EX-1108, 305; EX-1106, ¶15.

Schreiner1 provides an overview of Distributed Maple. EX-1108. Schreiner2 and Schreiner3 are referenced in Schreiner1 and provide further details regarding Distributed Maple. EX-1109, EX-1110. The Distributed Maple Code contains parts of the software described in Schreiner1, as well as related installation guides and readme files. EX-1106, ¶¶30-36. Schreiner1 expressly discusses the Distributed

- iii. **communicate a result of the first mathematical expression evaluation to a third node comprising a third hardware processor with a plurality of processing cores,**

The node contains program code, including the scheduler and dist.maple libraries, that communicates results of the first mathematical expression to the “third node” of the claim (e.g., the node in the bottom right of Figure 1 of Schreiner1). *See* sections X.A.3.f and X.A.3.g. The third node comprises a multi-core processor. *See* section X.A.3.f.i; EX-1105, ¶140.

- iv. **wherein the third node is configured to receive the result of mathematical expression evaluation...and communicate a result of the second mathematical expression evaluation to the first node;”**

The term “the first node” lacks antecedent basis and likely is a drafting error. There are two possible ways to construe “the first node.” EX-1105, ¶141.

If the “first node” is construed as “the second node” (i.e., the root node in Schreiner1), then the “third node” of the claim (e.g., the node in the bottom right of Figure 1 of Schreiner1) is configured to perform this limitation for the reasons already described for claim 26. *See* section X.A.3.g; EX-1105, ¶142.

If the “first node” is construed as “the computer cluster node,” then it would have been obvious to perform a sequence of tasks that result in the third node of the claim receiving a result from the computer cluster node, performing another mathematical expression evaluation, and providing the result of that evaluation to

the computer cluster node. This is because “[t]he execution of a task may take place on any machine connected to the distributed session,” and “[t]asks may freely create other tasks.” EX-1108, 312. Accordingly, in the normal course of Distributed Maple cluster operation, any node, including the third node, may receive results from any node, and may send results to any node. EX-1105, ¶143.

- f. “wherein the user connection interface is configured to return at least one result of mathematical expression evaluation to a user interface or a script; and”**

The user connection interface of the node is configured to return a result of the mathematical expression evaluation to the user interface of the second node (the root node). *See* section X.A.3.h; EX-1105, ¶144.

- g. “wherein the computer cluster node is configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to the single-node kernel.”**

All nodes, including the computer cluster node of this claim, are configured to perform this element. *See* section X.A.3.i. Any node, including the claimed node, may accept user instructions. For example, a node accepts a task comprising a Maple expression for evaluation, which may include “arbitrary Maple objects [that] may be passed as task arguments.” EX-1108, 312. Maple objects are

Nayak teaches that the DSP cluster is particularly well-suited for matrix operations: “[I]t is wise to implement computation intensive matrix operations on the DSP as they would take a longer time on general purpose processors.” EX-1136, section 5; EX-1105, ¶199.

A POSITA would have been motivated by Nayak to use Distributed Maple in conjunction with special purpose microprocessors comprising one or more DSP cores, as Nayak teaches that DSPs are particularly well suited for matrix operations, of which Distributed Maple performed many. Moreover, Nayak teaches cluster nodes with multiple DSP processing cores, disclosing the claim. EX-1136, section 4; EX-1105, ¶200.

XI. CONCLUSION

For these reasons, Petitioner requests *inter partes* review of the ’768 patent’s claims 26-27, 29, 35-37, and 39.

By: /Brent Yamashita/
Brent Yamashita

EXHIBIT V

**IN THE UNITED STATES DISTRICT COURT
FOR THE WESTERN DISTRICT OF TEXAS
WACO DIVISION**

PARKERVISION, INC.,
Plaintiff

-v-

**HISENSE CO., LTD., HISENSE
VISUAL TECHNOLOGY CO., LTD.**
Defendants

6-20-CV-00870-ADA

PARKERVISION, INC.,
Plaintiff

-v-

**TCL INDUSTRIES HOLDINGS CO.,
LTD., TCL ELECTRONICS
HOLDINGS LTD., SHENZHEN TCL
NEW TECHNOLOGY CO., LTD., TCL
KING ELECTRICAL APPLIANCES
(HUIZHOU) CO., LTD., TCL MOKA
INT'L LTD., TCL MOKA
MANUFACTURING S.A. DE C.V.**
Defendants

6-20-CV-00945-ADA

**SPECIAL MASTER’S REPORT AND RECOMMENDATION
REGARDING CLAIM CONSTRUCTION**

Before the Court are the Parties’ claim construction briefs: Defendants HiSense Co., Ltd. and HiSense Visual Technology Co., Ltd. (collectively “HiSense”) and TCL Industries Holdings Co., Ltd., TCL Electronics Holdings Ltd., Shenzhen TCL New Technology Co., Ltd., TCL King Electrical Appliances (Huizhou) Co., Ltd., TCL Moka Int’l Ltd., TCL Moka Manufacturing S.A. De C.V.’s (collectively “TCL”) Opening and Reply briefs (No. 6-20-cv-00870, ECF Nos. 33 and 42, respectively, and No. 6-20-cv-00945, ECF Nos. 33 and 40, respectively) (“Opening” and “Reply,” respectively) and Plaintiff ParkerVision, Inc. Response and Sur-Reply briefs (No. 6-20-cv-00870, ECF Nos. 40 and 44, respectively, and No. 6-20-cv-00945, ECF Nos. 38 and 42,

respectively) (“Response” and “Sur-Reply,” respectively). United States District Judge Alan D Albright referred these cases to the undersigned on October 25, 2021. No. 6-20-cv-00870, ECF No. 47 and No. 6-20-cv-00945, ECF No. 45. The undersigned provided preliminary constructions for the disputed terms the day before the hearing. No. 6-20-cv-00870, ECF No. 46 and No. 6-20-cv-00945, ECF No. 44. The undersigned held the *Markman* hearing on October 27, 2021. No. 6-20-cv-00870, ECF No. 48 and No. 6-20-cv-00945, ECF No. 46. During that hearing, the undersigned informed the Parties of the final recommended constructions for the disputed terms. *Id.* This Report does not alter any of those constructions.

I. BACKGROUND

Plaintiff asserts U.S. Patent Nos. 6,049,706, 6,266,518, 6,580,902, 7,110,444, 7,292,835, 8,588,725, 8,660,513, 9,118,528, 9,246,736, and 9,444,673. Plaintiff previously asserted these patents in the Western District of Texas against Intel in two cases (6-20-cv-00108, 6-20-cv-00562) and later against LG (6-21-cv-00520). Judge Albright held *Markman* hearings in the Intel cases on January 26, 2021 (-00108) and July 22, 2021 (-00562). Judge Albright previously construed Terms 3, 5–10, and 14–23 below in the prior Intel cases. 6-20-cv-00870,¹ ECF No. 51 at 3–9, 11–16.

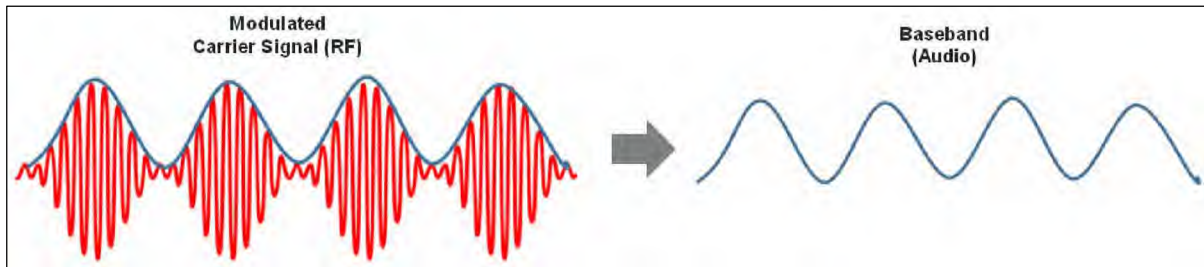
Judge Gilliland held a *Markman* hearing in *LG* case on May 10, 2022. No. 6-20-cv-00520, ECF No. 51. Judge Gilliland entered a *Markman* order and memorandum in support of his claim constructions on June 21, 2022. No. 6:21-cv-00520-ADA, 2022 WL 2240465 (W.D. Tex. June 21, 2022). In that order, Judge Gilliland provided his reasoning for his constructions for two terms (Term #1: “energy storage element” / “energy storage device” / “energy storage module” / “storage

¹ For simplicity, all references to the docket entries will be from the -00870 case.

element”/ “storage module” and Term #2: whether “cable modem” in U.S. Patent No. 7,292,835 Patent, Cl. 1 was limiting) and adopted Judge Albright’s constructions for 28 other terms (Terms #3 to #30). *Id.* Term #3 in this case corresponds to Term #1 in Judge Gilliland’s *Markman* order and memorandum in support thereof.

II. DESCRIPTION OF THE ASSERTED PATENTS

The Asserted Patents describe and claim systems for down-conversion of a modulated carrier signal. ’518 Patent at Abstract. Down-conversion is the process of recovering the baseband (audio) signal from the carrier signal after it has been transmitted to and received by the receiver. This process is referred to as “down-conversion” because a high frequency signal is being down-converted to a low frequency signal.



The Asserted Patents disclose at least two types of systems for down-conversion: (1) sample-and-hold (*i.e.*, voltage sampling) and (2) “energy transfer” (also known as “energy sampling”). The key difference between the two is that the former takes a small “sample” of the input signal while the latter takes a very large sample, *i.e.*, a large enough sample that a non-negligible amount of energy is transferred from the input signal. The following sub-sections describes each type of system, their respective operation, and compares them.

A. Circuit configuration of down-sampling systems: sample-and-hold and energy transfer.

the meaning of this claim term. *Nautilus*, 572 U.S. at 901; *Sonix Tech.*, 844 F.3d at 1377. As such, the undersigned recommends that this term is not indefinite and that it should be construed according to its plain-and-ordinary meaning.

N. Term #13: “frequency down-conversion module”

Term	Plaintiff’s Proposed Construction	TCL’s Proposed Construction
<p>#13: “frequency down-conversion module”</p> <p>U.S. Patent No. 7,110,444, Claims 2, 3</p> <p>Proposed by TCL</p>	<p>Plain-and-ordinary meaning</p>	<p>Subject to § 112, ¶ 6.</p> <p>Function: “to down-convert the input signal ... according to a [] control signal and output[] a [] down-converted signal.”</p> <p>Structure: an “aliasing module 2000” (blue) comprising at least one switch and one capacitor (Figures 20A and 20A-1).</p>

Judge Albright previously construed this in the *ParkerVision v. Intel* (6-20-cv-00108) case as being “Not subject to 35 U.S.C. § 112, ¶ 6” and “Plain-and-ordinary meaning.”

The undersigned first analyzes whether § 112, ¶ 6 applies. If so, the undersigned will then analyze what the function and corresponding structure is for this term.

The Parties’ Positions Regarding Whether § 112, ¶ 6 Applies:

TCL contends that § 112, ¶ 6 applies because (1) “module” is a well-known nonce word that operates as a substitute for means, (2) “frequency down-conversion” fails to connote sufficient structure, but merely designates the function of translating a frequency, and (3) that the specification describes this term in functional terms. Opening at 30.

Plaintiff contends that TCL makes the same arguments that the Court rejected in the 6-20-cv-00108 *Intel* case. Response at 33. Plaintiff contends that this term is not subject to § 112, ¶ 6 because (1) it is presumed not to be because of the absence of “means” and (2) the claims recite definite structure (*e.g.*, “frequency down-conversion” which receives an “input signal,” receives a “control signal,” and “outputs a first down-converted signal,” and “wherein said . . . down-conversion modules each comprise a switch and a storage element”). *Id.*

The Undersigned’s Analysis:

After reviewing the parties’ arguments and considering the applicable law, the undersigned agrees with Plaintiff that this term is not subject to § 112, ¶ 6 and should be construed according to its plain-and-ordinary meaning for the reasons that follow. **First**, there is no dispute that the term does not contain the words “means for” and that the presumption that the term is subject to § 112, ¶ 6 concomitantly does not apply. **Second**, a POSITA would understand that the claim term describes that the frequency translator “down-converts said input signal,” which would require using physical components including a switch and capacitor. A POSITA would also understand that the “control signal” recited in the claim language is used to control the electrical switch. The surrounding claim language also indicates that the frequency translator is a component within a physical device. For example, both claims describes that the frequency translator is a component within a wireless receiver. *See, e.g.*, ’444 Patent, Cl. 2, Lim. [a]. The undersigned finds that this physical structure is sufficient to prevent the application of § 112, ¶ 6. *TEK Glob.*, 920 F.3d at 786. **Third**, the surrounding claim language describes the input, output and connections of the delay module, which describes the structure of the claim term. *Apple*, 757 F.3d at 1299 (Fed. Cir. 2014) (“Structure may also be provided by describing the claim limitation’s operation, such as its

input, output, or connections.”). Here, the claim language describes that the input to the frequency translator is the input signal. *See, e.g.*, ’444 Patent, Cl. 2, Lim. [b] (“wherein said first frequency down-conversion module down-converts said input signal”). The claim language further describes that a control signal is an input into the frequency translator. *Id.* (“wherein said first frequency down-conversion module down-converts . . . according to a first control signal.”). The claim language also describes that a frequency translator outputs a down-converted signal into a subtractor. *See, e.g., Id.*, Cl. 2, Lim. [c]

Therefore, based on the foregoing, the undersigned recommends that this term is not subject to § 112, ¶ 6 and that it should be construed according to its plain-and-ordinary meaning.

O. Term #14: “Under-Sample” / “Under-Samples” / “Under-Sampling”

Term	Plaintiff’s Proposed Construction	Defendants’ Proposed Construction
<p>#14: “Under-Sample” / “Under-Samples” / “Under-Sampling”</p> <p>U.S. Patent No. 6,049,706, Cls. 1, 6, 7, 28; U.S. Patent No. 7,110,444, Cl. 2</p> <p>Proposed by ParkerVision</p>	<p>“sampling at an aliasing rate” or “sampling at less than or equal to twice the frequency of the input signal”</p>	<p>“sampling at less than or equal to twice the frequency of the input signal”</p>

Judge Albright previously construed this term this in the *ParkerVision v. Intel* (6-20-cv-00108 and 6-20-cv-00562) case as being “sampling at an aliasing rate” or “sampling at less than or equal to twice the frequency of the input signal.”

Second, the undersigned agrees with Plaintiff that collateral estoppel does not apply as “sampled energy” is a distinct term with a different meaning than “integrated energy.” In other words, because the dispute in the prior Middle District of Florida case is different than the instant cases, collateral estoppel does not apply.

Third, in addition to the above reasons, given that Judge Albright previously adopted Plaintiff’s construction, the undersigned believes it is better to recommend the same construction in order to align the recommended construction with Judge Albright’s previous construction.

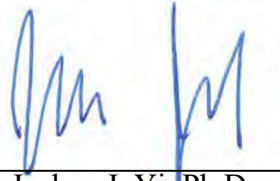
Therefore, based on the foregoing, the undersigned recommends that the construction of the term “a down-converted signal being generated from said sampled energy” should be “a lower frequency signal formed from sampled energy transferred from the electromagnetic signal when the switch module is closed and from sampled energy discharged from the storage module when the switch module is open.”

V. CONCLUSION

For the reasons described herein, the undersigned recommends that the Court adopt the following recommended constructions.

Pursuant to the Order Appointing Special Master (No. 6-20-cv-00870, ECF No. 47 and No. 6-20-cv-00945, ECF No. 45), the parties may, in a manner mirroring Federal Rule of Civil Procedure 72 and 28 U.S.C. § 636, file timely objections to any of the findings, conclusions, and recommendations contained in this Report.

SIGNED on the 29th day of August, 2022.

A handwritten signature in blue ink, appearing to read 'J. Yi', is positioned above a horizontal line.

Joshua J. Yi, Ph.D.

Term	Plaintiff's Proposed Construction	Defendants' Proposed Construction	Special Master's Recommended Construction
<p>#12: “establishing apertures” terms</p> <p>U.S. Patent No. 6,049,706, Claims 165, 107, 176, 187</p> <p>Proposed by TCL</p>	<p>Plain-and-ordinary meaning</p>	<p>Indefinite</p>	<p>Not indefinite. Plain-and-ordinary meaning.</p>
<p>#13: “frequency down-conversion module”</p> <p>U.S. Patent No. 7,110,444, Claims 2, 3</p> <p>Proposed by TCL</p>	<p>Plain-and-ordinary meaning</p>	<p>Subject to § 112, ¶ 6.</p> <p>Function: “to down-convert the input signal ... according to a [] control signal and output[] a [] down-converted signal.”</p> <p>Structure: an “aliasing module 2000” (blue) comprising at least one switch and one capacitor (Figures 20A and 20A-1).</p>	<p>Not subject to § 112, ¶ 6. Plain-and-ordinary meaning.</p>
<p>#14: “Under-Sample” / “Under-Samples” / “Under-Sampling”</p> <p>U.S. Patent No., Cls. 1, 6, 7, 28; U.S. Patent No. 7,110,444, Cl. 2</p> <p>Proposed by ParkerVision</p>	<p>“sampling at an aliasing rate” or “sampling at less than or equal to twice the frequency of the input signal”</p>	<p>“sampling at less than or equal to twice the frequency of the input signal”</p>	<p>“sampling at an aliasing rate” or “sampling at less than or equal to twice the frequency of the input signal”</p>

EXHIBIT W



US008082289B2

(12) **United States Patent**
Tannenbaum et al.

(10) **Patent No.:** **US 8,082,289 B2**
(45) **Date of Patent:** **Dec. 20, 2011**

(54) **CLUSTER COMPUTING SUPPORT FOR APPLICATION PROGRAMS**

(75) Inventors: **Zvi Tannenbaum**, Newport Beach, CA (US); **Dean E. Dager**, Huntington Beach, CA (US)

(73) Assignee: **Advanced Cluster Systems, Inc.**, Aliso Viejo, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1247 days.

(21) Appl. No.: **11/744,461**

(22) Filed: **May 4, 2007**

(65) **Prior Publication Data**

US 2007/0288935 A1 Dec. 13, 2007

Related U.S. Application Data

(60) Provisional application No. 60/813,738, filed on Jun. 13, 2006, provisional application No. 60/850,908, filed on Oct. 11, 2006.

(51) **Int. Cl.**
G06F 9/44 (2006.01)
G06F 15/16 (2006.01)

(52) **U.S. Cl.** **709/201**; 717/177; 719/320

(58) **Field of Classification Search** 709/201;
717/177; 719/320

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,423,046 A 6/1995 Nunnelley et al.
5,881,315 A 3/1999 Cohen
6,546,403 B1 4/2003 Carlson, Jr. et al.

6,782,537 B1 8/2004 Blackmore et al.
6,968,335 B2 11/2005 Bayliss et al.
7,136,924 B2 11/2006 Dager
7,249,357 B2 7/2007 Landman et al.
7,334,232 B2 2/2008 Jacobs et al.
2002/0049859 A1* 4/2002 Bruckert et al. 709/246
2003/0005266 A1 1/2003 Akkary et al.
2003/0051062 A1 3/2003 Circenis
2003/0135621 A1 7/2003 Romagnoli
2003/0195931 A1 10/2003 Dager
2003/0195938 A1 10/2003 Howard et al.
2004/0110209 A1 6/2004 Yokota et al.
2004/0157203 A1 8/2004 Dunk
2004/0252710 A1 12/2004 Jeter, Jr. et al.
2005/0021751 A1 1/2005 Block et al.

(Continued)

FOREIGN PATENT DOCUMENTS

JP 2002-117010 4/2002

OTHER PUBLICATIONS

International Search Report (Application No. PCT/US2007/076585) mailed Sep. 11, 2008.

(Continued)

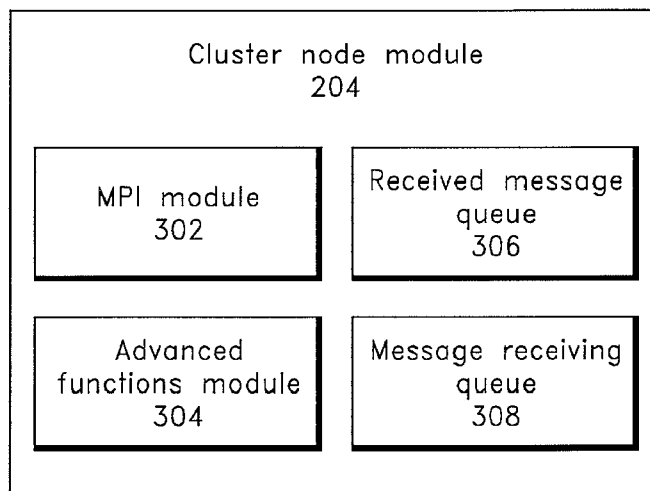
Primary Examiner — Larry Donaghue

(74) *Attorney, Agent, or Firm* — Knobbe Martens Olson & Bear, LLP

(57) **ABSTRACT**

A computer cluster system comprising a plurality of nodes and a software package comprising a user interface and a kernel for interpreting program code instructions is provided. In one embodiment, a cluster node module is configured to communicate with the kernel and other cluster node modules. The cluster node module accepts instructions from the user interface and interprets at least some of the instructions such that several cluster node modules in communication with one another and with a kernel can act as a computer cluster.

38 Claims, 5 Drawing Sheets



US 8,082,289 B2

Page 2

U.S. PATENT DOCUMENTS

2005/0038852	A1	2/2005	Howard	
2005/0060237	A1	3/2005	Barsness et al.	
2005/0076105	A1	4/2005	Keohane et al.	
2005/0108394	A1	5/2005	Braun et al.	
2005/0180095	A1	8/2005	Ellis	
2006/0053216	A1 *	3/2006	Deokar et al.	709/223
2006/0106931	A1	5/2006	Richoux	
2007/0073705	A1	3/2007	Gray	
2007/0288935	A1	12/2007	Tannenbaum et al.	
2008/0148244	A1	6/2008	Tannenbaum et al.	
2008/0250347	A1	10/2008	Gray et al.	
2008/0281997	A1	11/2008	Archer et al.	
2009/0222543	A1	9/2009	Tannenbaum et al.	

OTHER PUBLICATIONS

Mathematica Parallel Computing Toolkit Jan. 2005 pp. 1-90.
 Dauger et al. Plug-and Play Cluster Computing: High-Performance Computing for the Mainstream Apr. 2004 pp. 22-28.
 Dauger et al. "Plug-and-Play" Cluster Computing using Mac OS X 2003.
 Carns et al. An Evaluation of Message Passing Implementations on Beowulf Workstations Mar. 1999 pp. 41-54.
 Wolfram, Stephen: The Mathematica Book 5th Edition; Wolfram Research, Inc. 2003.

Maeder, Roman E.; Mathematica Parallel Computing Toolkit—Unleash the Power of Parallel Computing; Wolfram Research, Jan. 2005.

Tepeneu and Ida, MathGridLink—A bridge between Mathematica and the Grid, (online); Department of Computer Science, Graduate School of Systems and Information Engineering; University of Tsukuba, 2003; Retrieved from the Internet ,URL:<http://www.score.is.tsukuba.ac.jp/-ida/ida2004/Recent/bib/MathGridLink.pdf>.

(Abstract).
 International Search Report (Application No. PCT/US06/37275) mailed Sep. 24, 2007 (2 pages).

"gridMathematica 1.1: Grid Computing Gets a Speed Boost from Mathematica 5", The Mathematica Journal, vol. 9, No. 2, 2004.

Hamscher et al., "Evaluation of Job-Scheduling Strategies for grid Computing", LNCS: Lecture Notes in Computer Science, 2000, pp. 191-202.

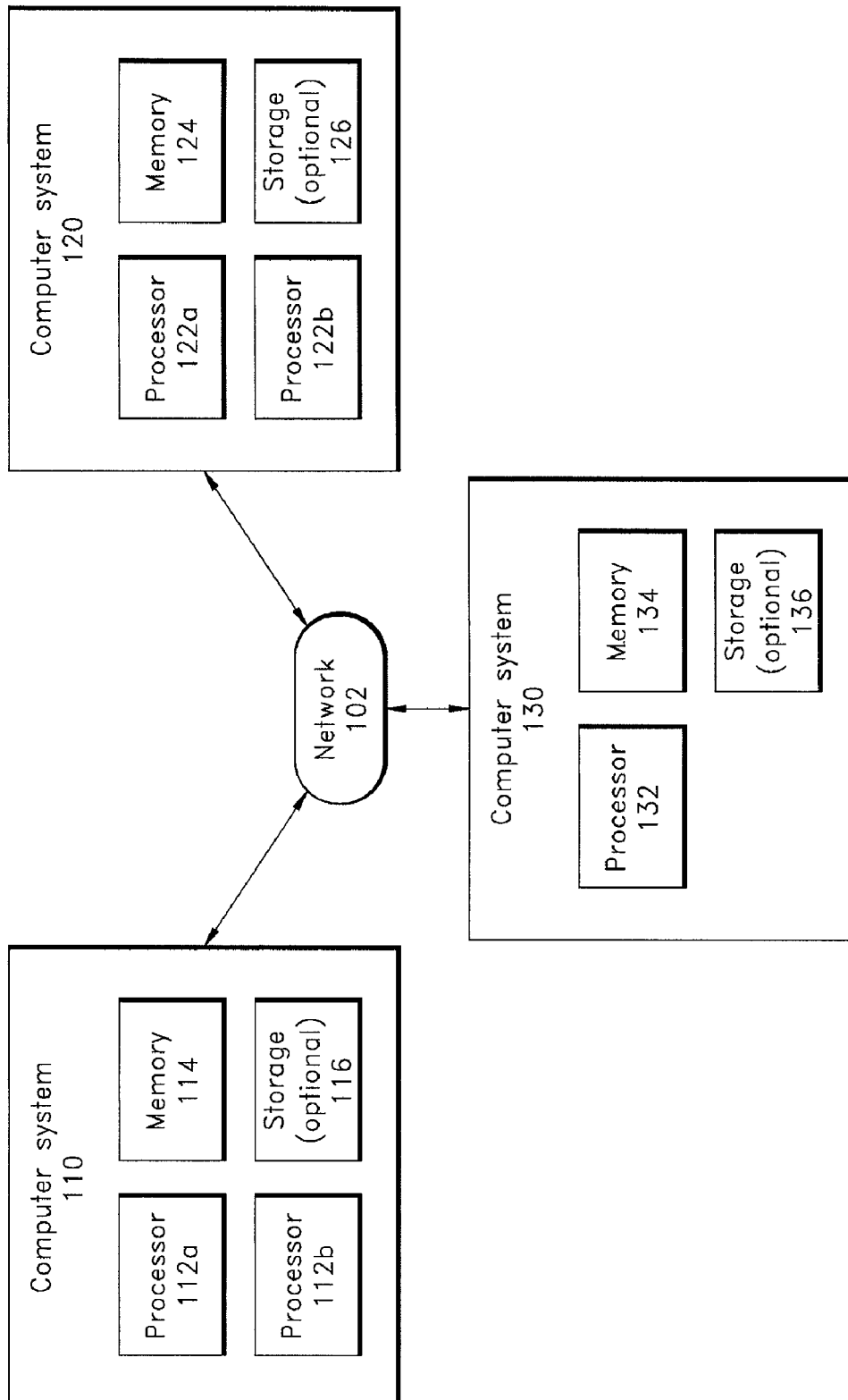
Jahanzeb et al., "Libra: a computational economy-based job scheduling system for clusters", Software Practice and Experience, Feb. 24, 2004, vol. 34, pp. 573-590.

Jain et al., "Data Clustering: A Review", ACM Computing Surveys, Sep. 1999, vol. 31, No. 3.

Office Action dated Jul. 11, 2011, U.S. Appl. No. 12/040,519, filed Feb. 28, 2008.

Website <http://wolfram.com/products/gridmathematica>, entitled Wolfram gridMathematica, printed on Oct. 3, 2007.

* cited by examiner

*FIG. 1*

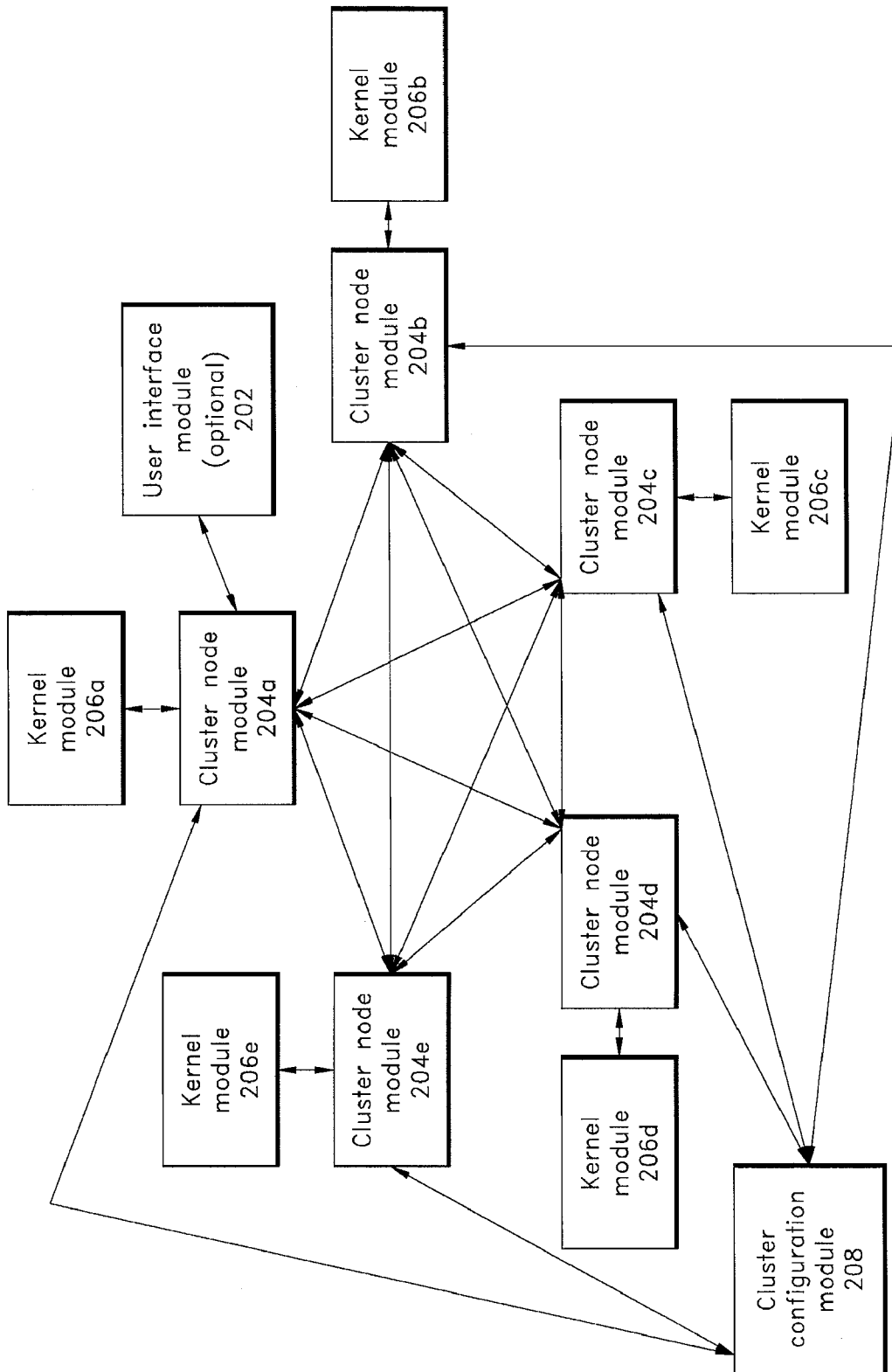


FIG. 2

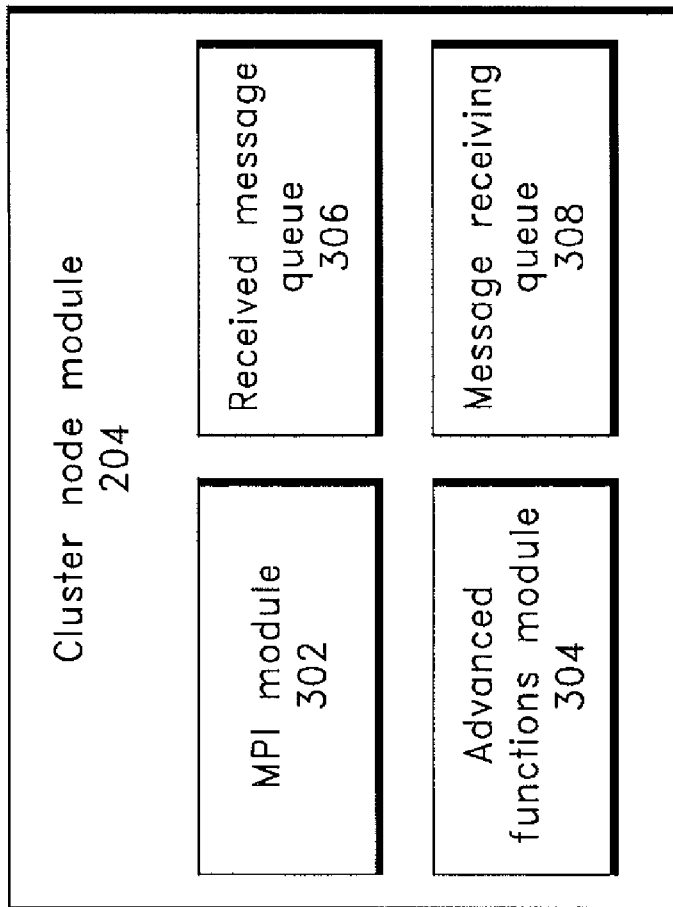
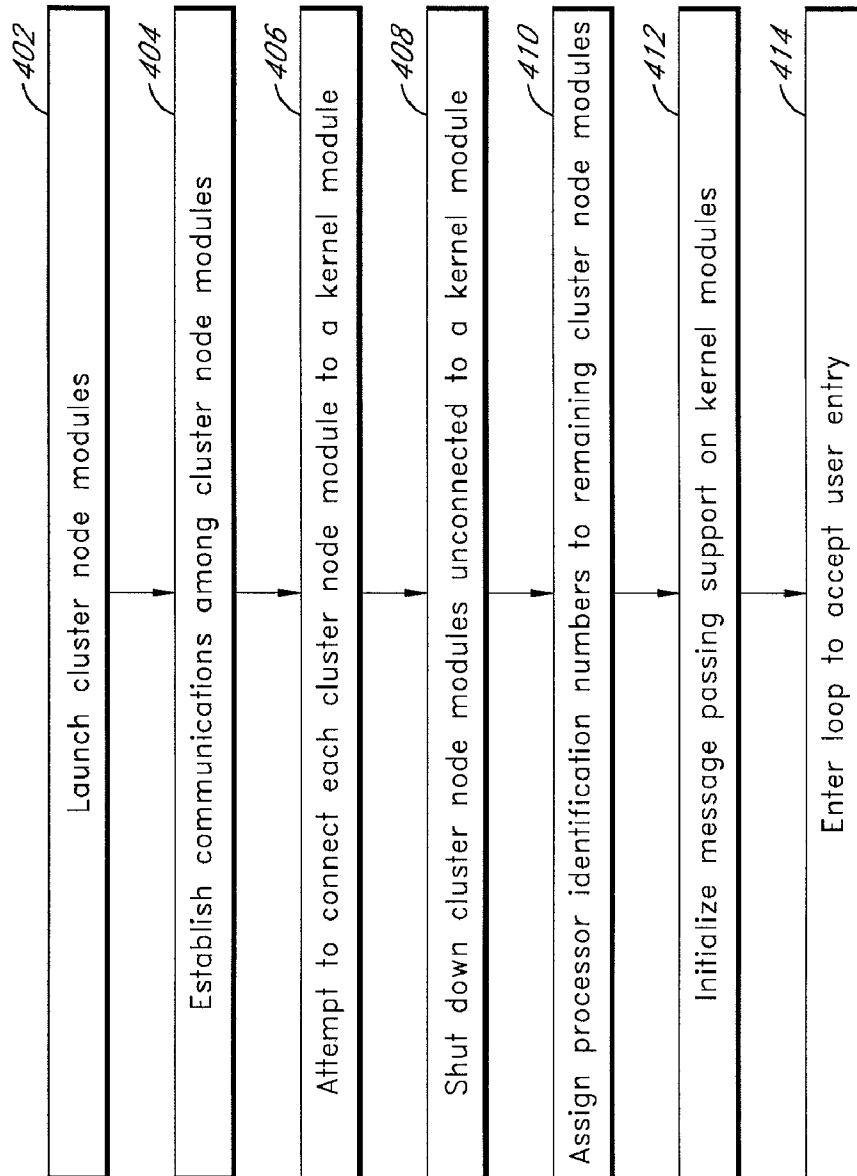
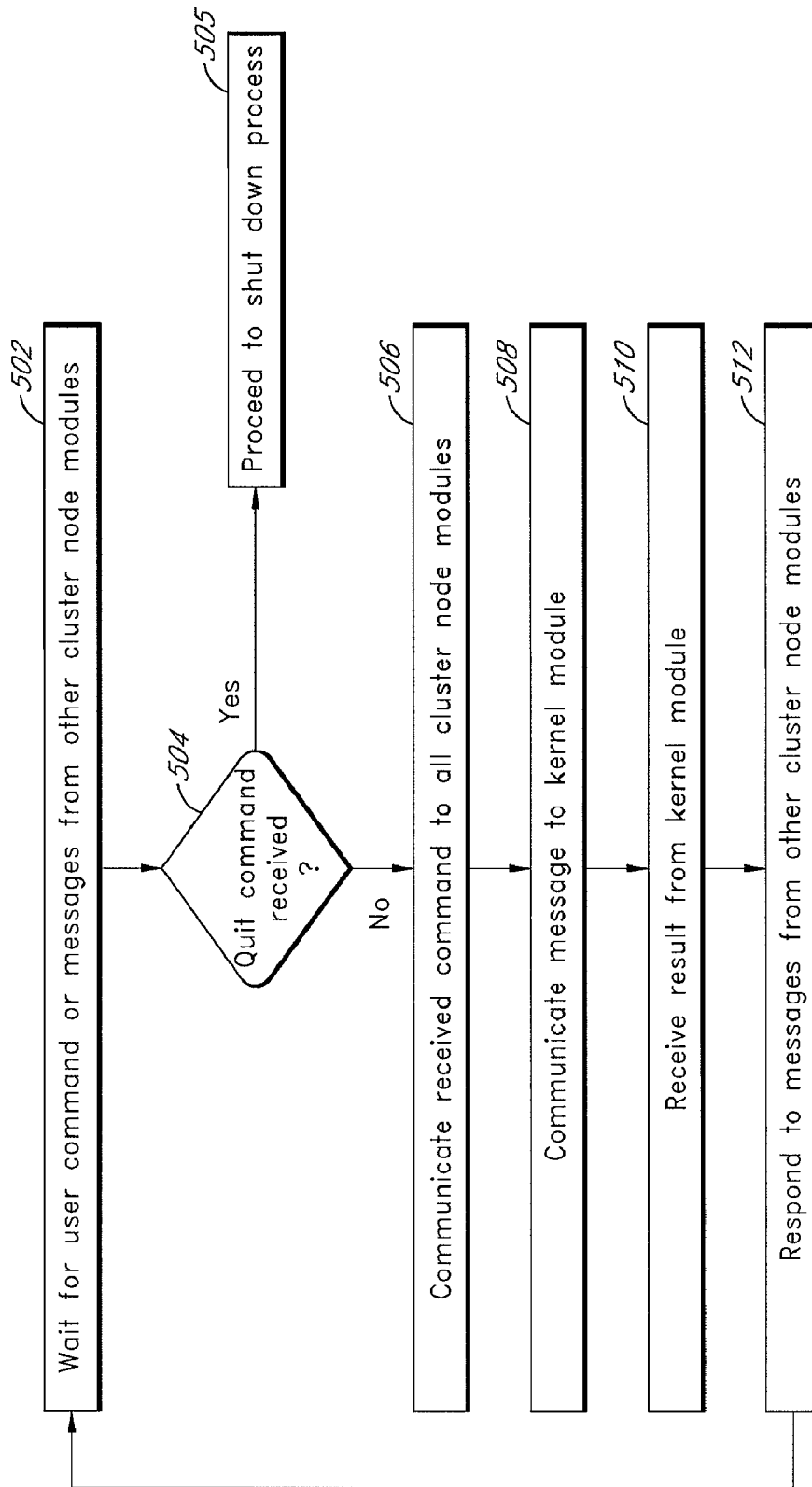


FIG. 3

*FIG. 4*

*FIG. 5*

US 8,082,289 B2

1

CLUSTER COMPUTING SUPPORT FOR
APPLICATION PROGRAMS

PRIORITY INFORMATION

This application claims priority to U.S. Provisional Patent Application No. 60/813,738, filed Jun. 13, 2006 and U.S. Provisional Patent Application No. 60/850,908, filed Oct. 11, 2006. The entirety of each of the above-referenced applications is hereby incorporated by reference and made part of this specification.

BACKGROUND

1. Field of the Disclosure

The present disclosure relates to the field of cluster computing generally and to systems and methods for adding cluster computing functionality to a computer program, in particular.

2. Description of the Related Art

Computer clusters include a group of two or more computers, microprocessors, and/or processor cores ("nodes") that intercommunicate so that the nodes can accomplish a task as though they were a single computer. Many computer application programs are not currently designed to benefit from advantages that computer clusters can offer, even though they may be running on a group of nodes that could act as a cluster. Some computer programs can run on only a single node because, for example, they are coded to perform tasks serially or because they are designed to recognize or send instructions to only a single node.

Some application programs include an interpreter that executes instructions provided to the program by a user, a script, or another source. Such an interpreter is sometimes called a "kernel" because, for example, the interpreter can manage at least some hardware resources of a computer system and/or can manage communications between those resources and software (for example, the provided instructions, which can include a high-level programming language). Some software programs include a kernel that is designed to communicate with a single node. An example of a software package that includes a kernel that is designed to communicate with a single node is Mathematica® from Wolfram Research, Inc. ("Mathematica"). Mathematica software packages from other vendors and other types of software can also include such a kernel.

A product known as gridMathematica, also from Wolfram Research, Inc., gives Mathematica the capability to perform a form of grid computing known as "distributed computing." Grid computers include a plurality of nodes that generally do not communicate with one another as peers. Distributed computing can be optimized for workloads that consist of many independent jobs or packets of work, which do not need to share data between the jobs during the computational process. Grid computers include at least one node known as a master node that manages a plurality of slave nodes or computational nodes. In gridMathematica, each of a plurality of kernels runs on a single node. One kernel is designated the master kernel, which handles all input, output, and scheduling of the other kernels (the computational kernels or slave kernels). Computational kernels receive commands and data only from the node running the master kernel. Each computational kernel performs its work independently of the other computational kernels and intermediate results of one job do not affect other jobs in progress on other nodes.

2

SUMMARY

Embodiments described herein have several features, no single one of which is solely responsible for their desirable attributes. Without limiting the scope of the invention as expressed by the claims, some of the advantageous features will now be discussed briefly.

Some embodiments described herein provide techniques for conveniently adding cluster computing functionality to a computer application. In one embodiment, a user of a software package may be able to achieve higher performance and/or higher availability from the software package by enabling the software to benefit from a plurality of nodes in a cluster. One embodiment allows a user to create applications, using a high-level language such as Mathematica, that are able to run on a computer cluster having supercomputer-like performance. One embodiment provides access to such high-performance computing through a Mathematica Front End, a command line interface, one or more high-level commands, or a programming language such as C or FORTRAN.

One embodiment adapts a software module designed to run on a single node, such as, for example, the Mathematica kernel, to support cluster computing, even when the software module is not designed to provide such support. One embodiment provides parallelization for an application program, even if no access to the program's source code is available. One embodiment adds and supports Message Passing Interface ("MPI") calls directly from within a user interface, such as, for example, the Mathematica programming environment. In one embodiment, MPI calls are added to or made available from an interactive programming environment, such as the Mathematica Front End.

One embodiment provides a computer cluster including a first processor, a second processor, and a third processor. The cluster includes at least one computer-readable medium in communication at least one of the first processor, the second processor, or the third processor. A first kernel resides in the at least one computer-readable medium and is configured to translate commands into code for execution on the first processor. A first cluster node module resides in the at least one computer-readable medium. The first cluster node module is configured to send commands to the first kernel and receives commands from a user interface. A second kernel resides in the at least one computer-readable medium. The second kernel is configured to translate commands into code for execution on the second processor. A second cluster node module resides in the at least one computer-readable medium. The second cluster node module is configured to send commands to the second kernel and communicates with the first cluster node module. A third kernel resides in the at least one computer-readable medium. The third kernel is configured to translate commands into code for execution on the third processor. A third cluster node module resides in the at least one computer-readable medium. The third cluster node module is configured to send commands to the third kernel and configured to communicate with the first cluster node module and the second cluster node module. The first cluster node module comprises a data structure in which messages originating from the second and third cluster node modules are stored.

Another embodiment provides a computer cluster that includes a plurality of nodes and a software package including a user interface and a single-node kernel for interpreting program code instructions. A cluster node module is configured to communicate with the single-node kernel and other cluster node modules. The cluster node module accepts instructions from the user interface and interprets at least some of the instructions such that several cluster node mod-

US 8,082,289 B2

3

ules in communication with one another act as a cluster. The cluster node module appears as a single-node kernel to the user interface. In one embodiment, the single-node kernel includes a Mathematica kernel. In some embodiments, the user interface can include at least one of a Mathematica front end or a command line. In some embodiments, the cluster node module includes a toolkit including library calls that implement at least a portion of MPI calls. In some embodiments, the cluster node module includes a toolkit including high-level cluster computing commands. In one embodiment, the cluster system can include a plurality of Macintosh® computers ("Macs"), Windows®-based personal computers ("PCs"), and/or Unix/Linux-based workstations.

A further embodiment provides a computer cluster including a plurality of nodes. Each node is configured to access a computer-readable medium comprising program code for a user interface and program code for a single-node kernel module configured to interpret user instructions. The cluster includes a plurality of cluster node modules. Each cluster node module is configured to communicate with a single-node kernel and with one or more other cluster node modules, to accept instructions from the user interface, and to interpret at least some of the user instructions such that the plurality of cluster node modules communicate with one another in order to act as a cluster. A communications network connects the nodes. One of the plurality of cluster node modules returns a result to the user interface.

Another embodiment provides a method of evaluating a command on a computer cluster. A command from at least one of a user interface or a script is communicated to one or more cluster node modules within the computer cluster. Each of the one or more cluster node modules communicates a message based on the command to a respective kernel module associated with the cluster node module. Each of the one or more cluster node modules receives a result from the respective kernel module associated with the cluster node module. At least one of the one or more cluster node modules responds to messages from other cluster node modules.

Another embodiment provides a computing system for executing Mathematica code on multiple nodes. The computing system includes a first node module in communication with a first Mathematica kernel executing on a first node, a second node module in communication with a second Mathematica kernel executing on a second node, and a third node module in communication with a third Mathematica kernel executing on a third node. The first node module, the second node module, and the third node module are configured to communicate with one another using a peer-to-peer architecture. In some embodiments, each of the first node module, the second node module, and third node module includes a data structure for maintaining messages originating from other node modules and a data structure for maintaining data specifying a location to which an message is expected to be received and an identifier for a node from which the message is expected to be sent.

BRIEF DESCRIPTION OF THE DRAWINGS

A general architecture that implements the various features are described with reference to the drawings. The drawings and the associated descriptions are provided to illustrate embodiments and not to limit the scope of the disclosure. Throughout the drawings, reference numbers are re-used to indicate correspondence between referenced elements.

FIG. 1 is a block diagram of one embodiment of a computer cluster.

4

FIG. 2 is a block diagram showing relationships between software modules running on one embodiment of a computer cluster.

FIG. 3 is a block diagram of one embodiment of a cluster node module.

FIG. 4 is a flow chart showing one embodiment of a cluster initialization process.

FIG. 5 is a flow chart showing one embodiment of the operation of a cluster node module.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

For purposes of illustration, some embodiments are described herein in the context of cluster computing with Mathematica software. The present disclosure is not limited to a single software program; the systems and methods can be used with other application software such as, for example, Maple®, MATLAB®, MathCAD®, Apple Shake®, Apple® Compressor, IDL®, other applications employing an interpreter or a kernel, Microsoft Excel®, Adobe After Effects®, Adobe Premiere®, Adobe Photoshop®, Apple Final Cut Pro®, and Apple iMovie®. Some figures and/or descriptions, however, relate to embodiments of computer clusters running Mathematica. The system can include a variety of uses, including but not limited to students, educators, scientists, engineers, mathematicians, researchers, and technicians. It is also recognized that in other embodiments, the systems and methods can be implemented as a single module and/or implemented in conjunction with a variety of other modules. Moreover, the specific implementations described herein are set forth in order to illustrate, and not to limit, the disclosure.

I. Overview

The cluster computing system described herein generally includes one or more computer systems connected to one another via a communications network or networks. The communications network can include one or more of a local area network ("LAN"), a wide area network ("WAN"), an intranet, the Internet, etc. In one embodiment, a computer system comprises one or more processors such as, for example, a microprocessor that can include one or more processing cores ("nodes"). The term "node" refers to a processing unit or subunit that is capable of single-threaded execution of code. The processors can be connected to one or more memory devices such as, for example, random access memory ("RAM"), and/or one or more optional storage devices such as, for example, a hard disk. Communications among the processors and such other devices may occur, for example, via one or more local buses of a computer system or via a LAN, a WAN, a storage area network ("SAN"), and/or any other communications network capable of carrying signals among computer system components. In one embodiment, one or more software modules such as kernels, run on nodes within the interconnected computer systems. In one embodiment, the kernels are designed to run on only a single node. In one embodiment, cluster node modules communicate with the kernels and with each other in order to implement cluster computing functionality.

FIG. 1 is a block diagram of one embodiment of a computer cluster 100 wherein computer systems 110, 120, 130 communicate with one another via a communications network 102. Network 102 includes one or more of a LAN, a WAN, a wireless network, an intranet, or the Internet. In one embodiment of the computer cluster, computer system 110 includes processors 112a, 112b, memory 114, and optional storage 116. Other computer systems 120, 130 can include similar devices, which generally communicate with one another

5

within a computer system over a local communications architecture such as a local bus (not shown). A computer system can include one or more processors, and each processor can contain one or more processor cores that are capable of single-threaded execution. Processor cores are generally independent microprocessors, but more than one can be included in a single chip package. Software code designed for single-threaded execution can generally run on one processor core at a time. For example, single-threaded software code typically does not benefit from multiple processor cores in a computer system.

FIG. 2 is a block diagram showing relationships among software modules running on one embodiment of a computer cluster 100. In the embodiment shown in FIG. 2, the kernel modules 206a-e are designed for single-threaded execution. For example, if each of the processors 112a, 112b, 122a, 122b, 132 shown in FIG. 1 includes only one processor core, two kernel modules (for example, kernel modules 206a, 206b) loaded into the memory 114 of computer system 110 could exploit at least some of the processing bandwidth of the two processors 112a, 112b. Similarly, two kernel modules 206c, 206d loaded into the memory 124 of computer system 120 could exploit at least some of the processing bandwidth of the two processors 122a, 122b. Likewise, the bandwidth of processor 132 of computer system 130 could be utilized by a single instance of a cluster node module 204e loaded into the computer system's memory 134.

In the embodiment shown in FIG. 2, each of the kernel modules 206a-e is in communication with a single cluster node module 204a-e, respectively. For example, the kernel module 206a is in communication with the cluster node module 204a, the kernel module 206b is in communication with the cluster node module 206b, and so forth. In one embodiment, one instance of a cluster node module 204a-e is loaded into a computer system's memory 114, 124, 134 for every instance of a kernel module 206a-e running on the system. As shown in FIG. 2, each of the cluster node modules 204a-e is in communication with each of the other cluster node modules 204a-e. For example, one cluster node module 204a is in communication with all of the other cluster node modules 204b-e. A cluster node module 204a may communicate with another cluster node module 204b via a local bus (not shown) when, for example, both cluster node modules 204a-b execute on processors 112a, 112b within the same computer system 110. A cluster node module 204a may also communicate with another cluster node module 204c over a communications network 102 when, for example, the cluster node modules 204a, c execute on processors 112a, 122a within different computer systems 110, 120.

As shown in FIG. 2, an optional user interface module 202 such as, for example, a Mathematica front end and/or a command line interface, can connect to a cluster node module 204a. The user interface module can run on the same computer system 110 and/or the same microprocessor 112a on which the cluster node module 204a runs. The cluster node modules 204a-e provide MPI calls and/or advanced cluster functions that implement cluster computing capability for the single-threaded kernel modules. The cluster node modules 204a-e are configured to look and behave like a kernel module 206a from the perspective of the user interface module 202. Similarly, the cluster node modules 204a-e are configured to look and behave like a user interface module 202 from the perspective of a kernel module 206a. The first cluster node module 204a is in communication with one or more other cluster node modules 204b, 204c, and so forth, each of which provides a set of MPI calls and/or advanced cluster com-

6

mands. In one embodiment, MPI may be used to send messages between nodes in a computer cluster.

Communications can occur between any two or more cluster node modules (for example, between a cluster node module 204a and another cluster node module 204c) and not just between "adjacent" kernels. Each of the cluster node modules 204a-e is in communication with respective kernel modules 206a-e. Thus, the cluster node module 204a communicates with the kernel module 206a. MPI calls and advanced cluster commands are used to parallelize program code received from an optional user interface module 208 and distribute tasks among the kernel modules 206a-e. The cluster node modules 204a-e provide communications among kernel modules 206a-e while the tasks are executing. Results of evaluations performed by kernel modules 206a-e are communicated back to the first cluster node module 204a via the cluster node modules 204a-e, which communicates them to the user interface module 208.

Intercommunication among kernel modules 206a-e during thread execution, which is made possible by cluster node modules 204a-e, provides advantages for addressing various types of mathematic and scientific problems, for example. Intercommunication provided by cluster computing permits exchange of information between nodes during the course of a parallel computation. Embodiments of the present disclosure provide such intercommunication for software programs such as Mathematica, while grid computing solutions can implement communication between only one master node and many slave nodes. Grid computing does not provide for communication between slave nodes during thread execution.

For purposes of providing an overview of some embodiments, certain aspects, advantages, benefits, and novel features of the invention are described herein. It is to be understood that not necessarily all such advantages or benefits can be achieved in accordance with any particular embodiment of the invention. Thus, for example, those skilled in the art will recognize that the invention can be embodied or carried out in a manner that achieves one advantage or group of advantages as taught herein without necessarily achieving other advantages or benefits as can be taught or suggested herein.

II. Computer Cluster 100

As shown in FIG. 1, one embodiment of a cluster system 100 includes computer systems 110, 120, 130 in communication with one another via a communications network 102. A first computer system 110 can include one or more processors 112a-b, a memory device 114, and an optional storage device 116. Similarly, a second computer system 120 can include one or more processors 122a-b, a memory device 124, and an optional storage device 126. Likewise, a third computer system 130 can include one or more processors 132, a memory device 134, and an optional storage device 136. Each of the computer systems 110, 120, 130 includes a network interface (not shown) for connecting to a communications network 102, which can include one or more of a LAN, a WAN, an intranet, a wireless network, and/or the Internet.

A. Computer System 110

In one embodiment, a first computer system 110 communicates with other computer systems 120, 130 via a network 102 as part of a computer cluster 100. In one embodiment, the computer system 110 is a personal computer, a workstation, a server, or a blade including one or more processors 112a-b, a memory device 114, an optional storage device 116, as well as a network interface module (not shown) for communications with the network 102.

US 8,082,289 B2

7

1. Processors **112a-b**

In one embodiment, the computer system **110** includes one or more processors **112a-b**. The processors **112a-b** can be one or more general purpose single-core or multi-core microprocessors such as, for example, a Pentium® processor, a Pentium® II processor, a Pentium® Pro processor, a Pentium® III processor, Pentium® 4 processor, a Core Duo® processor, a Core 2 Duo® processor, a Xeon® processor, an Itanium® processor, a Pentium® M processor, an x86 processor, an Athlon® processor, an 8051 processor, a MIPS® processor, a PowerPC® processor, an ALPHA® processor, etc. In addition, one or more of the processors **112a-b** can be a special purpose microprocessor such as a digital signal processor. The total number of processing cores (for example, processing units capable of single-threaded execution) within all processors **112a-b** in the computer system **110** corresponds to the number of nodes available in the computer system **110**. For example, if the processors **112a-b** were each Core 2 Duo® processors having two processing cores, computer system **110** would have four nodes in all. Each node can run one or more instances of a program module, such as a single-threaded kernel module.

2. Network Interface Module

The computer system **110** can also include a network interface module (not shown) that facilitates communication between the computer system **110** and other computer systems **120, 130** via the communications network **102**.

The network interface module can use a variety of network protocols. In one embodiment, the network interface module includes TCP/IP. However, it is to be appreciated that other types of network communication protocols such as, for example, Point-to-Point Protocol (“PPP”), Server Message Block (“SMB”), Serial Line Internet Protocol (“SLIP”), tunneling PPP, AppleTalk, etc., may also be used.

3. Memory **114** and Storage **116**

The computer system **110** can include memory **114**. Memory **114** can include, for example, processor cache memory (such as processor core-specific or cache memory shared by multiple processor cores), dynamic random-access memory (“DRAM”), static random-access memory (“SRAM”), or any other type of memory device capable of storing computer data, instructions, or program code. The computer system **110** can also include optional storage **116**. Storage **116** can include, for example, one or more hard disk drives, floppy disks, flash memory, magnetic storage media, CD-ROMs, DVDs, optical storage media, or any other type of storage device capable of storing computer data, instructions, and program code.

4. Computer System **110** Information

The computer system **110** may be used in connection with various operating systems such as: Microsoft® Windows® 3.X, Windows 95®, Windows 98®, Windows NT®, Windows 2000®, Windows XP®, Windows CE®, Palm Pilot OS, OS/2, Apple® MacOS®, MacOS X®, MacOS X Server®, Disk Operating System (DOS), UNIX, Linux®, VxWorks, or IBM® OS/2®, Sun OS, Solaris OS, IRIX OS operating systems, etc.

In one embodiment, the computer system **110** is a personal computer, a laptop computer, a Blackberry® device, a portable computing device, a server, a computer workstation, a local area network of individual computers, an interactive kiosk, a personal digital assistant, an interactive wireless communications device, a handheld computer, an embedded computing device, or the like.

As can be appreciated by one of ordinary skill in the art, the computer system **110** may include various sub-routines, procedures, definitional statements, and macros. Each of the

8

foregoing modules are typically separately compiled and linked into a single executable program. However, it is to be appreciated by one of ordinary skill in the art that the processes that are performed by selected ones of the modules may be arbitrarily redistributed to one of the other modules, combined together in a single module, made available in a shareable dynamic link library, or partitioned in any other logical way.

B. Computer System **120**

In one embodiment, a second computer system **120** communicates with other computer systems **110, 130** via a network **102** as part of a computer cluster **100**. In one embodiment, the computer system **120** is a personal computer, a workstation, a server, or a blade including one or more processors **122a-b**, a memory device **124**, an optional storage device **126**, as well as a network interface module (not shown) for communications with the network **102**.

1. Processors **122a-b**

In one embodiment, the computer system **120** includes one or more processors **122a-b**. The processors **122a-b** can be one or more general purpose single-core or multi-core microprocessors such as a Pentium® processor, a Pentium® II processor, a Pentium® Pro processor, a Pentium® III processor, Pentium® 4 processor, a Core Duo® processor, a Core 2 Duo® processor, a Xeon® processor, an Itanium® processor, a Pentium® M processor, an x86 processor, an Athlon® processor, an 8051 processor, a MIPS® processor, a PowerPC® processor, an ALPHA® processor, etc. In addition, the processors **122a-b** can be any special purpose microprocessors such as a digital signal processor. The total number of processing cores (for example, processing units capable of single-threaded execution) within all processors **122a-b** in the computer system **120** corresponds to the number of nodes available in the computer system **120**. For example, if the processors **122a-b** were each Core 2 Duo® processors having two processing cores, computer system **120** would have four nodes in all. Each node can run one or more instances of a program module, such as a single-threaded kernel module.

2. Network Interface Module

The computer system **120** can also include a network interface module (not shown) that facilitates communication between the computer system **120** and other computer systems **110, 130** via the communications network **102**.

The network interface module can use a variety of network protocols. In one embodiment, the network interface module includes TCP/IP. However, it is to be appreciated that other types of network communication protocols such as, for example, Point-to-Point Protocol (“PPP”), Server Message Block (“SMB”), Serial Line Internet Protocol (“SLIP”), tunneling PPP, AppleTalk, etc., may also be used.

3. Memory **124** and Storage **126**

The computer system **120** can include memory **124**. Memory **124** can include, for example, processor cache memory (such as processor core-specific or cache memory shared by multiple processor cores), dynamic random-access memory (“DRAM”), static random-access memory (“SRAM”), or any other type of memory device capable of storing computer data, instructions, or program code. The computer system **120** can also include optional storage **126**. Storage **126** can include, for example, one or more hard disk drives, floppy disks, flash memory, magnetic storage media, CD-ROMs, DVDs, optical storage media, or any other type of storage device capable of storing computer data, instructions, and program code.

4. Computer System **120** Information

The computer system **120** may be used in connection with various operating systems such as: Microsoft® Windows®

3.X, Windows 95®, Windows 98®, Windows NT®, Windows 2000®, Windows XP®, Windows CE®, Palm Pilot OS, OS/2, Apple® MacOS®, MacOS X®, MacOS X Server®, Disk Operating System (DOS), UNIX, Linux®, VxWorks, or IBM® OS/2®, Sun OS, Solaris OS, IRIX OS operating systems, etc.

In one embodiment, the computer system **120** is a personal computer, a laptop computer, a Blackberry® device, a portable computing device, a server, a computer workstation, a local area network of individual computers, an interactive kiosk, a personal digital assistant, an interactive wireless communications device, a handheld computer, an embedded computing device, or the like.

As can be appreciated by one of ordinary skill in the art, the computer system **120** may include various sub-routines, procedures, definitional statements, and macros. Each of the foregoing modules are typically separately compiled and linked into a single executable program. However, it is to be appreciated by one of ordinary skill in the art that the processes that are performed by selected ones of the modules may be arbitrarily redistributed to one of the other modules, combined together in a single module, made available in a shareable dynamic link library, or partitioned in any other logical way.

C. Computer System **130**

In one embodiment, a third computer system **130** communicates with other computer systems **110**, **120** via a network **102** as part of a computer cluster **100**. In one embodiment, the computer system **130** is a personal computer, a workstation, a server, or a blade including one or more processors **132**, a memory device **134**, an optional storage device **136**, as well as a network interface module (not shown) for communications with the network **102**.

1. Processors **112a-b**

In one embodiment, the computer system **130** includes a processor **132**. The processor **132** can be a general purpose single-core or multi-core microprocessors such as a Pentium® processor, a Pentium® II processor, a Pentium® Pro processor, a Pentium® III processor, Pentium® 4 processor, a Core Duo® processor, a Core 2 Duo® processor, a Xeon® processor, an Itanium® processor, a Pentium® M processor, an x86 processor, an Athlon® processor, an 8051 processor, a MIPS® processor, a PowerPC® processor, or an ALPHAS processor. In addition, the processor **132** can be any special purpose microprocessor such as a digital signal processor. The total number of processing cores (for example, processing units capable of single-threaded execution) within processor **132** in the computer system **130** corresponds to the number of nodes available in the computer system **130**. For example, if the processor **132** was a Core 2 Duo® processor having two processing cores, the computer system **130** would have two nodes. Each node can run one or more instances of a program module, such as a single-threaded kernel module.

2. Network Interface Module

The computer system **130** can also include a network interface module (not shown) that facilitates communication between the computer system **130** and other computer systems **110**, **120** via the communications network **102**.

The network interface module can use a variety of network protocols. In one embodiment, the network interface module includes TCP/IP. However, it is to be appreciated that other types of network communication protocols such as, for example, Point-to-Point Protocol ("PPP"), Server Message Block ("SMB"), Serial Line Internet Protocol ("SLIP"), tunneling PPP, AppleTalk, etc., may also be used.

3. Memory **134** and Storage **136**

The computer system **130** can include memory **134**. Memory **134** can include, for example, processor cache memory (such as processor core-specific or cache memory shared by multiple processor cores), dynamic random-access memory ("DRAM"), static random-access memory ("SRAM"), or any other type of memory device capable of storing computer data, instructions, or program code. The computer system **130** can also include optional storage **136**. Storage **136** can include, for example, one or more hard disk drives, floppy disks, flash memory, magnetic storage media, CD-ROMs, DVDs, optical storage media, or any other type of storage device capable of storing computer data, instructions, and program code.

4. Computer System **130** Information

The computer system **130** may be used in connection with various operating systems such as: Microsoft® Windows® 3.X, Windows 95®, Windows 98®, Windows NT®, Windows 2000®, Windows XP®, Windows CE®, Palm Pilot OS, OS/2, Apple® MacOS®, MacOS X®, MacOS X Server®, Disk Operating System (DOS), UNIX, Linux®, VxWorks, or IBM® OS/2®, Sun OS, Solaris OS, IRIX OS operating systems, etc.

In one embodiment, the computer system **130** is a personal computer, a laptop computer, a Blackberry® device, a portable computing device, a server, a computer workstation, a local area network of individual computers, an interactive kiosk, a personal digital assistant, an interactive wireless communications device, a handheld computer, an embedded computing device, or the like.

As can be appreciated by one of ordinary skill in the art, the computer system **130** may include various sub-routines, procedures, definitional statements, and macros. Each of the foregoing modules are typically separately compiled and linked into a single executable program. However, it is to be appreciated by one of ordinary skill in the art that the processes that are performed by selected ones of the modules may be arbitrarily redistributed to one of the other modules, combined together in a single module, made available in a shareable dynamic link library, or partitioned in any other logical way.

E. Communications Network **102**

In one embodiment, computer systems **110**, **120**, **130** are in communication with one another via a communications network **102**.

The communications network **102** may include one or more of any type of electronically connected group of computers including, for instance, the following networks: a virtual private network, a public Internet, a private Internet, a secure Internet, a private network, a public network, a value-added network, a wired network, a wireless network, an intranet, etc. In addition, the connectivity to the network can be, for example, a modem, Ethernet (IEEE 802.3), Gigabit Ethernet, 10-Gigabit Ethernet, Token Ring (IEEE 802.5), Fiber Distributed Datalink Interface (FDDI), Frame Relay, Infini-Band, Myrinet, Asynchronous Transfer Mode (ATM), or another interface. The communications network **102** may connect to the computer systems **110**, **120**, **130**, for example, by use of a modem or by use of a network interface card that resides in each of the systems.

In addition, the same or different communications networks **102** may be used to facilitate communication between the first computer system **110** and the second computer system **120**, between the first computer system **110** and the third computer system **130**, and between the second computer system **120** and the third computer system **130**.

III. Software Modules

As shown in FIGS. 1 and 2, one embodiment of a cluster system **100** includes a user interface module **202** that is able to access a plurality of kernel modules **206a-e** by communicating with a first cluster node module **204a**. User interface module can be stored in a memory **114**, **124**, **134** while running, for example, and/or can be stored in a storage device **116**, **126**, **136**. The first cluster node module **204a** is in communication with each of the other cluster node modules **204b-e**. The kernel modules **206a-e** can reside in the memory of one or more computer systems on which they run. For example, the memory **114** of the first computer system **110** can store instances of kernel modules **206a-b**, the memory **124** of the second computer system **120** can store instances of kernel modules **206c-d**, and the memory **134** of the third computer system **130** can store an instance of kernel module **206e**. The kernel modules **206a-e**, which include single-threaded program code, are each associated with one of the processors **112a**, **112b**, **122a**, **122b**, **132**. A cluster configuration module stored on one or more of the computer systems **110**, **120**, **130** or on a remote computer system, for example, can establish communication with the cluster node modules **204a-e**. In one embodiment, communication between the cluster configuration module **208** and the cluster node modules **204a-e** initializes the cluster node modules **204a-e** to provide cluster computing support for the computer cluster **100**.

A. Cluster Node Module **204**

In one embodiment, the cluster node modules **204a-e** provide a way for many kernel modules **206a-e** such as, for example, Mathematica kernels, running on a computer cluster **100** to communicate with one another. A cluster node module **204** can include at least a portion of an application programming interface (“API”) known as the Message-Passing Interface (“MPI”), which is used in several supercomputer and cluster installations. A network of connections (for example, the arrows shown in FIG. 2) between the cluster node modules **204a-e** can be implemented using a communications network **102**, such as, for example, TCP/IP over Ethernet, but the connections could also occur over any other type of network or local computer bus.

A cluster node module **204** can use an application-specific toolkit or interface such as, for example, Mathematica’s MathLink, Add-Ons, or packets, to interact with an application. Normally used to connect a Mathematica kernel to a user interface known as the Mathematica Front End or other Mathematica kernels, MathLink is a bidirectional protocol to send “packets” containing messages, commands, or data between any of these entities. MathLink does not allow direct cluster computing-like simultaneous communication between Mathematica kernels during execution of a command or thread. MathLink is also not designed to perform multiple simultaneous network connections. In some embodiments, a cluster node module **204** can use an application-specific toolkit such as, for example, MathLink, for connections between entities on the same computer.

When speaking about procedures or actions on a cluster or other parallel computer, not all actions happen in sequential order, nor are they required to. For example, a parallel code, as opposed to a single-processor code of the classic “Turing machine” model, has multiple copies of the parallel code running across the cluster, typically one for each processor (or “processing element” or “core”). Such parallel code is written in such a way that different instances of the same code can communicate, collaborate, and coordinate work with each other. Multiple instances of these codes can run at the same time in parallel.

If the count of the code instances is an integer N , each instance of code execution can be labeled **0** through $N-1$. For example, a computer cluster can include N connected computers, each containing a processor. The first has cluster node module **0** connected with kernel module **0** running on processor **0**. The next is cluster node module **1** and kernel module **1**, on processor **1**, and so forth for each of the N connected computers. Some steps of their procedure are collaborative, and some steps are independent. Even though these entities are not necessarily in lock-step, they do follow a pattern of initialization, main loop behavior (for example, cluster node module operation), and shut down.

In contrast, a parallel computing toolkit (PCT) that is provided as part of the gridMathematica software package does not provide a means for instances of the same code running on different nodes to communicate, collaborate, or coordinate work among the instances. The PCT provides commands that connect Mathematica kernels in a master-slave relationship rather than a peer-to-peer relationship as enabled by some embodiments disclosed herein. A computer cluster having peer-to-peer node architecture performs computations that can be more efficient, easier to design, and/or more reliable than similar computations performed on grid computers having master-slave node architecture. Moreover, the nature of some computations may not allow a programmer to harness multi-node processing power on systems that employ master-slave node architecture.

FIG. 3 shows one embodiment of a cluster node module **204** implementing MPI calls and advanced MPI functions. In the embodiment shown in FIG. 3, cluster node module **204** includes MPI module **302**, advanced functions module **304**, received message queue **306**, and message receiving queue **308**.

1. MPI module **302**

In one embodiment, the cluster node module **204** includes an MPI module **302**. The MPI module **302** can include program code for one or more of at least five kinds of MPI instructions or calls. Selected constants, instructions, and/or calls that can be implemented by the MPI module **302** are as follows:

MPI Constants

Node identifiers are used to send messages to nodes or receive messages from them. In MPI, this is accomplished by assigning each node a unique integer ($\$IdProc$) starting with **0**. This data, with a knowledge of the total count ($\$NProc$), makes it possible to programmatically divide any measurable entity.

TABLE A

Constant	Description
$\$IdProc$	The identification number of the current processor
$\$NProc$	The number of processors in the current cluster
$\$mpiCommWorld$	The communicator world of the entire cluster (see MPI Communicator routines, below)
$mpiCommWorld$	The default communicator world for the high-level routines.

Basic MPI Calls

In one embodiment, the MPI module **302** can include basic MPI calls such as, for example, relatively low-level routines that map MPI calls that are commonly used in other languages (such as C and Fortran), so that such calls can be available directly from the Mathematica user interface **204**. In some embodiments, basic MPI calls include calls that send data, equations, formulas, and/or other expressions.

US 8,082,289 B2

13

Simply sending expressions from one node to another is possible with these most basic MPI calls. One node can call to send an expression while the other calls a corresponding routine to receive the sent expression. Because it is possible that the receiver has not yet called `mpiRecv` even if the message has left the sending node, completion of `mpiSend` is not a confirmation that it has been received.

TABLE B

Call	Description
<code>mpiSend[expr, target, comm, tag]</code>	Sends an expression <code>expr</code> to a node with the ID target in the communicator would comm, waiting until that expression has left this kernel
<code>mpiRecv [expr, target, comm, tag]</code>	Receives an expression into <code>expr</code> from a node with the ID target in the communicator world comm, waiting until the expression has arrived
<code>mpiSendRecv[sendexpr, dest, recvexpr, source, comm]</code>	Simultaneously sends the expression <code>sendexpr</code> to the node with the ID target and receives an expression into <code>recvexpr</code> from the node with the ID source in the communicator world comm, waiting until both operations have returned.

Asynchronous MPI Calls

Asynchronous calls make it possible for the kernel to do work while communications are proceeding simultaneously. It is also possible that another node may not be able to send or receive data yet, allowing one kernel to continue working while waiting.

TABLE C

Call	Description
<code>mpiISend[expr, target, comm, tag, req]</code>	Sends an expression <code>expr</code> to a processor with the ID target in the communicator world comm, returning immediately. It can be balanced with calls to <code>mpiTest[req]</code> until <code>mpiTest[req]</code> returns True.
<code>mpiIRecv[expr, target, comm, tag, req]</code>	Receives an expression <code>expr</code> from a processor with the ID target in the communicator world comm, returning immediately. It can be balanced with calls to <code>mpiTest[req]</code> until <code>mpiTest[req]</code> returns True. The <code>expr</code> is not safe to access until <code>mpiTest[req]</code> returns True.
<code>mpiTest[req]</code>	Completes asynchronous behavior of <code>mpiISend</code> and <code>mpiIRecv</code>
<code>mpWait[req]</code>	Calls <code>mpiTest</code> until it returns True.
<code>mpiWaitall[reglist]</code>	Calls <code>mpiWait</code> all on every element of <code>reglist</code>
<code>mpiWaitany[reqlist]</code>	Calls <code>mpiTest</code> on each element of <code>reqlist</code> until one of them returns True

The `mpiISend[]` command can be called from within a kernel module **206** (for example, a Mathematica kernel). It creates a packet containing the Mathematica expression to be sent as payload and where the expression should be sent. The packet itself is destined only for its local cluster node module. Once received by its local cluster node module, this packet is decoded and its payload is forwarded on to the cluster node module specified in the packet.

The `mpiIRecv[]` command can also be called from within a kernel module **206**. It creates a packet specifying where it expects to receive an expression and from which processor this expression is expected. Once received by its local cluster node module, this packet is decoded and its contents are stored in a message receiving queue (MRQ) **308** (FIG. 3).

The `mpiTest[]` command can be called from within a kernel module **206**. It creates a packet specifying which mes-

14

sage to test for completion, then waits for a reply expression to evaluate. Once received by the kernel module's associated cluster node module **204**, this packet is decoded and its message specifier is used to search for any matching expressions listed as completed in its received message queue (RMQ) **306**. If such completed expressions are found, it is sent to its local kernel module as part of the reply in `mpiTest[]`. The kernel module receives this reply expression and evaluates it, which updates the kernel module's variables as needed.

Other MPI calls are built on the fundamental calls `mpiISend`, `mpiIRecv`, and `mpiTest`. For example, `mpiBcast`, a broadcast, creates instructions to send information from the broadcast processor to all the others, while the other processors perform a `Recv`. Similarly, high-level calls of the toolkit can be built on top of the collection of MPI calls.

Collective MPI Calls

In one embodiment, the MPI module **302** can include program code for implementing collective MPI calls (for example, calls that provide basic multi-node data movement across nodes). Collective MPI calls can include broadcasts, gathers, transpose, and other vector and matrix operations, for example. Collective calls can also provide commonly used mechanisms to send expressions between groups of nodes.

TABLE D

Call	Description
<code>mpiBcast[expr, root, comm]</code>	Performs a broadcast of <code>expr</code> from the root processor to all the others in the communicator world comm. An expression is expected to be supplied by the root processor, while all the others expect <code>expr</code> to be overwritten by the incoming expression.
<code>mpiGather[sendexpr, recvexpr, root, comm]</code>	All processors (including root) in the communicator comm send their expression in <code>sendexpr</code> to the root processor, which produces a list of these expressions, in the order according to comm, in <code>recvexpr</code> . On the processors that are not root, <code>recvexpr</code> is ignored.
<code>mpiAllgather[sendexpr, recvexpr, comm]</code>	All processors in the communicator comm send their expression in <code>sendexpr</code> , which are organized into a list of these expressions, in the order according to comm, in <code>recvexpr</code> on all processors in comm.
<code>mpiScatter[sendexpr, recvexpr, root, comm]</code>	Processor root partitions the list in <code>sendexpr</code> into equal parts (if possible) and places each piece in <code>recvexpr</code> on all the processors (including root) in the communicator world comm, according to the order and size of comm.
<code>mpiAlltoall[sendexpr, recvexpr, comm]</code>	Each processor sends equal parts of the list in <code>sendexpr</code> to all other processors in the communicator world comm, which each collects from all other processors are organized into the order according to comm.

In one embodiment, the MPI module **302** includes program code for implementing parallel sums and other reduction operations on data stored across many nodes. MPI module **302** can also include program code for implementing simple parallel input/output calls (for example, calls that allow cluster system **200** to load and store objects that are located on a plurality of nodes).

TABLE E

Call	Description
<code>mpiReduce[sendexpr, recvexpr, operation, root, comm]</code>	Performs a collective reduction operation between expressions on all processors in the communicator world comm for every element in the list in <code>sendexpr</code> returning

US 8,082,289 B2

15

TABLE E-continued

Call	Description
	the resulting list in recvexpr on the processor with the ID root.
mpiAllreduce[sendexpr, recvexpr, operation, comm]	Performs a collective reduction operation between expressions on all processors in the communicator world comm for every element in the list in sendexpr returning the resulting list in recvexpr on every processor.
mpiReduceScatter[sendexpr, recvexpr, operation, comm]	Performs a collective reduction operation between expressions on all processors in the communicator world comm for every element in the list in sendexpr, partitioning the resulting list into pieces for each processor's recvexpr.

These additional collective calls perform operations that reduce the data in parallel. The operation argument can be one of the constants below.

TABLE F

Constant	Description
mpiSum	Specifies that all the elements on different processors be added together in a reduction call
mpiMax	Specifies that the maximum of all the elements on different processors be chosen in a reduction call
mpiMin	Specifies that the minimum of all the elements on different processors be chosen in a reduction call

MPI Communicator Calls

In one embodiment, the MPI module **302** includes program code for implementing communicator world calls (for example, calls that would allow subsets of nodes to operate as if they were a sub-cluster). Communicators organize groups of nodes into user-defined subsets. The communicator values returned by mpiCommSplit[] can be used in other NPI calls instead of mpiCommWorld.

TABLE G

Call	Description
mpiCommSize[comm]	Returns the number of processors within the communicator comm
mpiCommRank[comm]	Returns the rank of this processor in the communicator comm
mpiCommDup[comm]	Returns a duplicate communicator of the communicator comm
mpiCommSplit[comm, color, key]	Creates a new communicator into several disjoint subsets each identified by color. The sort order within each subset is first by key, second according to the ordering in the previous communicator. Processors not meant to participate in any new communicator indicates this by passing the constant mpiUndefined. The corresponding communicator is returned to each calling processor.
mpiCommMap[comm] mpiCommMap[comm, target]	Returns the mapping of the communicator comm to the processor indexed according to \$mpiCommWorld. Adding a second argument returns just the ID of the processor with the ID target in the communicator comm.
mpiCommFree[comm]	Frees the communicator comm

16

Other MPI Support Calls

Other calls that provide common functions include:

TABLE H

Call	Description
mpiWtime[]	Provides wall-dock time since some fixed time in the past. There is no guarantee that this time will read the same on all processors.
10 mpWtick[] MaxByElement[in]	Returns the time resolution of mpiWtime[]. For every nth element of each list of the list in, chooses the maximum according to Max[], and returns the result as one list. Used in the mpiMax reduction operation.
15 MinByElement[in]	For every nth element of each list of the list in, chooses the minimum according to Min[], and returns the result as one list. Used in the mpiMin reduction operation.

2. Advanced Functions Module **304**

In one embodiment, the cluster node module **204** includes an advanced functions module **304**. The advanced functions module **304** can include program code that provides a toolkit of functions inconvenient or impractical to do with MPI instructions and calls implemented by the MPI module **302**. The advanced functions module **304** can rely at least partially on calls and instructions implemented by the MPI module **302** in the implementation of advanced functions. In one embodiment, the advanced functions module **304** includes a custom set of directives or functions. In an alternative embodiment, the advanced functions module **304** intercepts normal Mathematica language and converts it to one or more functions optimized for cluster execution. Such an embodiment can be easier for users familiar with Mathematica functions to use but can also complicate a program debugging process. Some functions implemented by the advanced functions module **304** can simplify operations difficult or complex to set up using parallel computing. Several examples of such functions that can be implemented by the advanced functions module **304** are shown below.

Built on the MPI calls, the calls that are described below provide commonly used communication patterns or parallel versions of Mathematica features. Unless otherwise specified, these are executed in the communicator mpiCommWorld, whose default is \$mpiCommWorld, but can be changed to a valid communicator at run time.

Common Divide-and-Conquer Parallel Evaluation

In one embodiment, the advanced functions module **304** includes functions providing for basic parallelization such as, for example, routines that would perform the same operations on many data elements or inputs, stored on many nodes. These functions can be compared to parallelized for-loops and the like. The following calls address simple parallelization of common tasks. In the call descriptions, "expr" refers to an expression, and "loopspec" refers to a set of rules that determine how the expression is evaluated. In some embodiments, the advanced functions module **304** supports at least three forms of loopspec, including {var, count}, where the call iterates the variable var from 1 to the integer count; {var, start, stop}, where the call iterates the variable var every integer from start to stop; and {var, start, stop, increment}, where the call iterates the variable var from start adding increment for each iteration until var exceeds stop, allowing var to be a non-integer.

US 8,082,289 B2

17

TABLE I

Call	Description
ParallelDo[expr, loopspec]	Like Do[] except that it evaluates expr across the cluster, rather than on just one processor. The rules for how expr is evaluated is specified in loopspec, like in Do[].
ParallelFunctionToList[f, count]	Evaluates the function f[i] from 1 to count, but across the cluster, and returns these results in a list. The third argument has it gather this list into the processor whose ID is root.
ParallelTable[expr, loopspec]	Like Table[] except that it evaluates expr across the cluster, rather than on just one processor, returning the locally evaluated portion. The third argument has it gather this table in to the processor whose ID is root.
ParallelTable[expr, loopspec, root]	Like f[inputs] except that it evaluates f on a subset of inputs scattered across the cluster from processor root and gathered back to root.
ParallelFunction[f, inputs, root]	Like Nintegrate[] except that it evaluates a numerical integration of expr over domains partitioned into the number of processors in the cluster, then returns the sum. The third argument has each
ParallelNintegrate[expr, loopspec]	
ParallelNintegrate[expr, loopspec, digits]	

TABLE I-continued

Call	Description
	numerical integration execute with at least that many digits of precision.

Guard-Cell Management

In one embodiment, the advanced functions module **304** includes functions providing for guard-cell operations such as, for example, routines that perform nearest-neighbor communications to maintain edges of local arrays in any number of dimensions (optimized for 1-, 2-, and/or 3-D). Typically the space of a problem is divided into partitions. Often, however, neighboring edges of each partition can interact, so a

18

“guard cell” is inserted on both edges as a substitute for the neighboring data. Thus the space a processor sees is two elements wider than the actual space for which the processor is responsible. EdgeCell helps maintain these guard cells.

TABLE J

Call	Description
EdgeCell[list]	Copies the second element of list to the last element of the left processor and the second-to-last element of list to the first element of the right processor while simultaneously receiving the same from its neighbors.

Matrix and Vector Manipulation

The advanced functions module **304** can also include functions providing for linear algebra operations such as, for example, parallelized versions of basic linear algebra on structures partitioned on many nodes. Such linear algebra operations can reorganize data as needed to perform matrix and vector multiplication or other operations such as determinants, trace, and the like. Matrices are partitioned and stored in processors across the cluster. These calls manipulate these matrices in common ways.

TABLE K

Call	Description
ParallelTranspose[matrix]	Like Transpose[] except that it transposes matrix that is in fact represented across the cluster, rather than on just one processor. It returns the portion of the transposed matrix meant for that processor.
ParallelProduct[matrix, vector]	Evaluates the product of matrix and vector, as it would on one processor, except that matrix is represented across the cluster.
ParallelDimensions[matrix]	Like Dimensions[] except that matrix is represented across the cluster, rather than on just one processor. It returns a list of each dimension.
ParallelTr[matrix]	Like Tr[] except that the matrix is represented across the cluster, rather than on just one processor. It returns the trace of this matrix.
ParallelIdentity[rank]	Like Identity[], it generates a new identity matrix, except that the matrix is represented across the cluster, rather than on just one processor. It returns the portion of the new matrix for this processor.
ParallelOuter[f, vector1, vector2]	Like Outer[f, vector1, vector2] except that the answer becomes a matrix represented across the cluster, rather than on just one processor. It returns the portion of the new matrix for this processor.
ParallelInverse[matrix]	Like Inverse[] except that the matrix is represented across the cluster, rather than on just one processor. It returns the inverse of the matrix.

Element Management

In one embodiment, the advanced functions module **304** includes element management operations. For example, a large bin of elements or particles cut up in space across the nodes may need to migrate from node to node based on rules or criteria (such as their spatial coordinate). Such operations would migrate the data from one node to another. Besides the divide-and-conquer approach, a list of elements can also be partitioned in arbitrary ways. This is useful if elements need to be organized or sorted onto multiple processors. For example, particles of a system may drift out of the space of one processor into another, so their data would need to be redistributed periodically.

US 8,082,289 B2

19

TABLE L

Call	Description
ElementManage[list, switch]	Selects which elements of list will be sent to which processors according to the function switch[] is evaluated on each element of list. If switch is a function, switch[] should return the ID of the processor that element should be sent. If switch is an integer, the call assumes that each elements is itself a list, whose first element is a number ranging from 0 to the passed argument. This call returns a list of the elements, from any processor, that is switch selected for this processor.
ElementManage[list]	Each element of list can be a list of two elements, the first being the ID of the processor where the element should be sent, while the second is arbitrary data to send. This call returns those list elements, from any and all processors, whose first element is this processors ID in a list. This call is used internally by the two-argument version of ElementManage[].

20

Fourier Transform

In one embodiment, the advanced functions module **304** includes program code for implementing large-scale parallel fast Fourier transforms (“FFTs”). For example, such functions can perform FFTs in one, two, and/or three dimensions on large amounts of data that are not stored on one node and that are instead stored on many nodes. Fourier transforms of very large arrays can be difficult to manage, not the least of which is the memory requirements. Parallelizing the Fourier transform makes it possible to make use of all the memory available on the entire cluster, making it possible to manipulate problem sizes that no one processor could possibly do alone.

TABLE M

Call	Description
ParallelFourier[list]	Like Fourier[] except that list is a two- or three-dimensional list represented across the cluster, like for matrices, above. It returns the portion of the Fourier-transformed array meant for that processor.

Parallel Disk I/O

In one embodiment, the advanced functions module **304** includes parallel disk input and output calls. For example, data may need to be read in and out of the cluster in such a way that the data is distributed across the cluster evenly. The calls in the following table enable the saving data from one or more processors to storage and the retrieval data from storage.

TABLE N

Call	Description
ParallelPut[expr, filename]	Puts expr into the file with the name filename in order on processor 0. The third argument specifies that the file be written on the processor whose ID is root. The fourth uses the communicator world comm.
ParallelPut[expr, filename, root]	
ParallelPut[expr, filename, root, comm]	
ParallelGet[filename]	Reads and returns data from the file with the name filename on processor 0 partitioned into each processor on the cluster. The second argument specifies that the file is to be read on the processor whose ID is root. The third uses the communicator world comm.
ParallelGet[filename, root]	
ParallelGet[filename, root, comm]	
ParallelBinaryPut[expr, type, filename]	Puts expr into the file with the binary format type with the name filename in order on processor 0. The fourth argument specifies that the file be written on the processor whose ID is root. The fifth uses the communicator world comm.
ParallelBinaryPut[expr, filename, root]	
ParallelBinaryPut[expr, filename, root, comm]	
ParallelBinaryGet[type, filename]	Reads and returns data in the binary format type from the file with the name filename on processor 0 partitioned into each processor on the cluster. The third argument specifies that the file is to be read on the processor whose ID is root. The fourth uses the communicator world comm.
ParallelBinaryGet[type, filename, root]	
ParallelBinaryGet[type, filename, root, comm]	
ParallelGetPerProcessor[expr, filename]	Puts expr into the file with the name filename in order on processor 0, one line per processor. The third argument specifies that the file be written on the processor whose ID is root. The fourth uses the communicator world comm.
ParallelGetPerProcessor[filename, root]	
ParallelGetPerProcessor[filename, root, comm]	
ParallelGetPerProcessor[filename]	Reads and returns data from the file with the name filename on processor 0, one line for each processor. The second argument specifies that the file is to be read on the processor whose ID is root. The third uses the communicator world comm.
ParallelGetPerProcessor[filename, root]	

TABLE N-continued

Call	Description
ParallelGetPerProcessr [filename, root, comm]	

Automatic Load Balancing

Some function calls can take an inconsistent amount of processing time to complete. For example, in Mathematica, the call f[20] could in general take much longer to evaluate than f[19]. Moreover, if one or more processors within the cluster are of different speeds (for example, if some operate at a core frequency of 2.6 GHz while other operate at less than one 1 GHz), one processor may finish a task sooner than another processor.

In some embodiments, the advanced functions module 304 includes a call that can improve the operation of the computer cluster 100 in such situations. In some embodiments, the root processor assigns a small subset of the possible calls for a function to each processor on the cluster 100. Whichever processor returns its results first is assigned a second small subset of the possible calls. The root processor will continue to assign small subsets of the possible calls as results are received until an evaluation is complete. The order in which the processors finish can vary every time an expression is evaluated, but the root processor will continue assigning additional work to processors as they become available.

In one illustrative example, there are 4 processors and f[1] to f[100] to evaluate. One could implement this by assigning f[1], f[2], f[3], f[4] to each of processors 0 (the root can assign to oneself) through 3. If the f[2] result came back first, then processor 1 would be assigned f[5]. If the f[4] result is returned next, f[6] would be assigned to processor 3. The assignments continue until all results are calculated. The results are organized for output back to the user.

In alternative embodiments, the subsets of possible calls can be assigned in any order, rather than sequentially, or in batches (for example, f[1], f[5], f[9] assigned to processor 1, etc.). Also, the subsets could be organized by delegation. For example, one processor node may not necessarily be in direct control of the other processors. Instead, a large subset could be assigned to a processor, which would in turn assign subsets of its work to other processors. The result would create a hierarchy of assignments like a vast army.

TABLE O

LoadBalanceFunctionToList[f, count]	Evaluates the function f[i] from 1 to count, but across the cluster
LoadBalanceFunctionToList[f, count, root]	using load-balancing techniques, and returns these results in a list. The third argument has it gather this list into the processor whose ID is root.

3. Received Message Queue 306

In one embodiment, the cluster node module 204 includes a received message queue 306. The received message queue 306 includes a data structure for storing messages received from other cluster node modules. Related data pertaining to the messages received, such as whether an expression has been completed, may also be stored in the received message queue 306. The received message queue 306 may include a queue and/or another type of data structure such as, for example, a stack, a linked list, an array, a tree, etc.

4. Message Receiving Queue 308

In one embodiment, the cluster node module 204 includes a message receiving queue 308. The message receiving queue 308 includes a data structure for storing information about the location to which an expression is expected to be sent and the processor from which the expression is expected. The message receiving queue 308 may include a queue and/or another type of data structure such as, for example, a stack, a linked list, an array, a tree, etc.

B. Cluster Configuration Module 208

Cluster configuration module 208 includes program code for initializing a plurality of cluster node modules to add cluster computing support to computer systems 110, 120, 130. U.S. Pat. No. 7,136,924, issued to Dauger (the “’924 patent”), the entirety of which is hereby incorporated by reference and made a part of this specification, discloses a method and system for parallel operation and control of computer clusters. One method generally includes obtaining one or more personal computers having an operating system with discoverable of network services. In some embodiments, the method includes obtaining one or more processors or processor cores on which a kernel module can run. As described in the ’924 patent, a cluster node control and interface (CNCI) group of software applications is copied to each node. When the CNCI applications are running on a node, the cluster configuration module 208 can permit a cluster node module 204, in combination with a kernel module 206, to use the node’s processing resources to perform a parallel computation task as part of a computer cluster. The cluster configuration module 208 allows extensive automation of the cluster creation process in connection with the present disclosure.

C. User Interface Module 202

In some embodiments, computer cluster 100 includes a user interface module 202, such as, for example a Mathematica Front End or a command line interface, that includes program code for a kernel module 206 to provide graphical output, accept graphical input, and provide other methods of user communication that a graphical user interface or a command-line interface provides. To support a user interface module 202, the behavior of a cluster node module 204a is altered in some embodiments. Rather than sending output to and accepting input from the user directly, the user interface module 202 activates the cluster node module 204a to which it is connected and specifies parameters to form a connection, such as a MathLink connection, between the cluster node module 204a and the user interface module 202. The user interface module’s activation of the cluster node module 204a can initiate the execution of instructions to activate the remaining cluster node modules 204b-e on the cluster and to complete the sequence to start all kernel modules 206a-e on the cluster. Packets from the user interface module 202, normally intended for a kernel module 206a, are accepted by the cluster node module 204a as a user command. Output from the kernel module 206a associated with the cluster node module 204a can be forwarded back to the user interface module 202 for display to a user. Any of the cluster node modules 204a-e can be configured to communicate with a user interface module 202.

D. Kernel Module 206

A kernel module 206 typically includes program code for interpreting high-level code, commands, and/or instructions supplied by a user or a script into low-level code, such as, for example, machine language or assembly language. In one embodiment, each cluster node module 204a-e is connected to all other cluster node modules, while each kernel module 206a-e is allocated and connected only to one cluster node module 204. In one embodiment, there is one cluster node module-kernel module pair per processor. For example, in an embodiment of a computer cluster 100 including single-processor computer systems, each cluster node module-kernel module pair could reside on a single-processor computer. If a computer contains multiple processors or processing cores, it may contain multiple cluster node module-kernel module pairs, but the pairs can still communicate over the cluster node module's network connections.

IV. Cluster Computing Methods

In one embodiment, the computer cluster 100 includes a cluster initialization process, a method of cluster node module operation, and a cluster shut down process.

A. Cluster Initialization Process

In one embodiment, a cluster configuration module 202 initializes one or more cluster node modules 204 in order to provide cluster computing support to one or more kernel modules 206, as shown in FIG. 4.

At 402, cluster node modules are launched on the computer cluster 100. In one embodiment, the cluster node module 204a running on a first processor 112a (for example, where the user is located) accesses the other processors 112b, 122a-b, 132 on the computer cluster 100 via the cluster configuration module 208 to launch cluster node modules 204b-e onto the entire cluster. In an alternative embodiment, the cluster configuration module 208 searches for processors 112a-b, 122a-b, 132 connected to one another via communications network 102 and launches cluster node modules 204a-e on each of the processors 112a-b, 122a-b, 132.

The cluster node modules 204a-e establish communication with one another at 404. In one embodiment, each of the cluster node modules 204a-e establish direct connections using the MPI_Init command with other cluster node modules 204a-e launched on the computer cluster 100 by the cluster configuration module 208.

At 406, each cluster node module 204 attempts to connect to a kernel module 206. In one embodiment, each instance of the cluster node modules 204a-e locates, launches, and connects with a local kernel module via MathLink connections and/or similar connection tools, for example, built into the kernel module 206.

At 408, the cluster node modules 204 that are unconnected to a kernel module 206 are shut down. In one embodiment, each cluster node module 204 determines whether the local kernel module cannot be found or connected to. In one embodiment, each cluster node module 204 reports the failure to connect to a kernel module 206 to the other cluster node modules on computer cluster 100 and quits.

Processor identification numbers are assigned to the remaining cluster node modules 204 at 410. In one embodiment, each remaining cluster node module 204 calculates the total number of active processors (N) and determines identification numbers describing the remaining subset of active cluster node modules 204a-e and kernel modules 206a-e. This new set of cluster node module-kernel module pairs may be numbered 0 through N-1, for example.

Message passing support is initialized on the kernel modules 206a-e at 412. In one embodiment, each cluster node

module 204 supplies initialization code (for example, Mathematica initialization code) to the local kernel module 206 to support message passing.

Finally, at 414, the cluster node modules 204a-e enter a loop to accept user entry. In one embodiment, a main loop (for example, a cluster operation loop) begins execution after the cluster node module 204a on the first processor 112a returns to user control while each of the other cluster node modules 204 waits for messages from all other cluster node modules 204a-e connected to the network 102.

The initialization process creates a structure enabling a way for the kernel modules 206a-e to send messages to one another. In some embodiments, any kernel module can send data to and receive data from any other kernel module within the cluster when initialization is complete. The cluster node module creates an illusion that a kernel module is communicating directly with the other kernel modules. The initialization process can create a relationship among kernel modules on a computer cluster 100 such as the one shown by way of example in FIG. 2.

B. Cluster Node Module Operation

In one embodiment, a cluster node module 204 implements cluster computing support for a kernel module 206 during a main loop, as shown in FIG. 5.

At 502, cluster node modules 204 wait for user commands or messages from other cluster node modules. In one embodiment, the cluster node module 204a connected to the user interface module 202 waits for a user command, while the other cluster node modules 204b-e continue checking for messages.

Once a command or message is received, the method proceeds to 504. At 504, the cluster node module 204a determines whether the message received is a quit command. If a quit command is received, the cluster node module 204a exits the loop and proceeds to a cluster node module shut down process at 505. If the message received is not a quit command, the process continues to 506.

At 506, received commands are communicated to all cluster node modules 204a-e on the computer cluster 100. In one embodiment, when a user enters a command in the user interface module 202, the cluster node module 204a connected to the user interface module 202 submits the user command to all other cluster node modules 204b-e in the computer cluster 100. The user commands can be simple (for example, "1+1"), but can also be entire subroutines and sequences of code (such as, for example, Mathematica code), including calls to MPI from within the user interface module 202 (for example, the Mathematica Front End) to perform message passing between kernel modules 206a-e (for example, Mathematica kernels). These include the fundamental MPI calls, which are implemented using specially identified messages between a cluster node module 204 and its local kernel module 206.

The message (or user command) is communicated to the kernel modules 206a-e at 508. In one embodiment, the cluster node module 204a connected to the user interface module 202 submits the user command to the kernel module 206a to which it is connected. Each of the other cluster node modules 204b-e, after receiving the message, submits the command to the respective kernel module 206b-e to which it is connected.

At 510, a cluster node module 204 receives a result from a kernel module 206. In one embodiment, once the kernel module 206 completes its evaluation, it returns the kernel module's output to the cluster node module 204 to which it is connected. Depending on the nature of the result from the kernel module, the cluster node module 204 can report the result to a local computer system or pass the result as a

US 8,082,289 B2

25

message to another cluster node module **204**. For example, the cluster node module **204a** running on the first processor **112a** reports the output on its local computer system **110**. For example, on the first processor **112a**, cluster node module **204a** only directly reports the output of kernel module **206a**.

Messages from other cluster node modules **204** are responded to at **512**. In one embodiment, each cluster node module (for example, the cluster node module **204a**) checks for and responds to messages from other cluster node modules **204b-e** and from the kernel module **206a** repeatedly until those are exhausted. In one embodiment, output messages from the kernel module **206** are forwarded to output on the local computer system. Messages from other cluster node modules **204** are forwarded to a received message queue **306** ("RMQ"). Data from each entry in the message receiving queue **308** ("MRQ") is matched with entries in the RMQ **306** (see, for example, description of the `mpiRecv` call, above). If found, data from the MRQ **308** are combined into those in the RMQ **306** and marked as "completed" (see, for example, description of the `mpiTest` call, above). This process provides the peer-to-peer behavior of the cluster node modules **204a-e**. Via this mechanism, code running within multiple, simultaneously running kernel modules (for example, Mathematica kernels) can interact on a pair-wise or collective basis, performing calculations, processing, or other work on a scale larger and/or faster than one kernel could have done alone. In this manner, user-entered instructions and data specifying what work will be done via user commands can be executed more quickly and/or reliably. Once responding to messages has completed, the process returns to **502**.

C. Cluster Shut Down Process

In one embodiment, a computer cluster **100** includes a procedure to shut down the system. If the operation process (or main loop) on the cluster node module **204a** connected to the user interface module **202** detects a "Quit" or "Exit" command or otherwise receives a message from the user indicating a shut down, the sequence to shut down the cluster node modules **204a-e** and the kernel modules **206a-e** is activated. In one embodiment, the cluster node module **204a** connected to the user interface module **202** sends a quit message to all other cluster node modules **204b-e**. Each cluster node module **204** forwards the quit command to its local kernel module **206**. Once its Mathematica kernel has quit, each cluster node module **204** proceeds to tear down its communication network with other cluster node modules (for example, see description of the `MPI_Finalize` command, above). At the conclusion of the process, each cluster node module **204** exits execution.

V. Example Operation

For purposes of illustration, sample scenarios are discussed in which the computer cluster system is used in operation. In these sample scenarios, examples of Mathematica code are given, and descriptions of how the code would be executed by a cluster system are provided.

Basic MPI

Fundamental data available to each node includes the node's identification number and total processor count.

```
In[1]:={$IdProc, $NProc}
Out[1]:={0, 2}
```

The first element should be unique for each processor, while the second is generally the same for all. Processor **0** can see what other values are using a collective (see below) communications call such as `mpiGather`.

26

```
In[2]:=mpiGather[{$IdProc, $NProc},list,0]; list
Out[2]:={{0, 2}, {1, 2}}
```

Peer-to-Peer MPI

The `mpiSend` and `mpiRecv` commands make possible basic message passing, but one needs to define which processor to target. The following defines a new variable, `targetProc`, so that each pair of processors will point to each other.

```
In[3]:=targetProc=If[1==Mod[$IdProc, 2],$IdProc-1,
Out[3]:=1
```

In this example, the even processors target its "right" processor, while the odd ones point its "left." For example, if the processors were lined up in a row and numbered in order, every even-numbered processor would pair with the processor following it in the line, and every odd-numbered processor would pair with the processor preceding it. Then a message can be sent:

```
In[4]:=If[1==Mod[$IdProc,
2],mpiSend[N[Pi,22],targetProc, mpiCommWorld,d],
mpiRecv[a,targetProc,mpiCommWorld,d]]
```

The `If` statement causes the processors to evaluate different code: the odd processor sends 22 digits of π , while the even receives that message. Note that these MPI calls return nothing. The received message is in the variable `a`:

```
In[5]:=a
Out[5]:=3.1415926535897932384626
In[6]:=Clear[a]
```

The variable `a` on the odd processors would have no definition. Moreover, if `$NProc` is 8, processor **3** sent π to processor **2**, processor **5** sent π to processor **4**, and so on. These messages were not sent through processor **0**, but they communicated on their own.

The `mpiSend` and `mpiRecv` commands have a letter "I" to indicate asynchronous behavior, making it possible to do other work while messages are being sent and received, or if the other processor is busy. So, the above example could be done asynchronously:

```
In[7]:=If[1==Mod[$IdProc, 2],mpiSend[N[Pi,22],targetProc,
mpiCommWorld,d,e], mpiRecv[a,targetProc,
mpiCommWorld,d,e]]
```

The variable `e` has important data identifying the message, and `mpiTest[e]` can return `True` before the expressions are to be accessed. At this point, many other evaluations can be performed. Then, one can check using `mpiTest` when the data is needed:

```
In[29]:=mpiTest[e]
Out[29]:=True
In[30]:=a
Out[30]:=3.1415926535897932384626
In[31]:=Clear[a,e]
```

The `mpiWait[e]` command could have also have been used, which does not return until `mpiTest[e]` returns `True`. The power of using these peer-to-peer calls is that it becomes possible to construct any message-passing pattern for any problem.

US 8,082,289 B2

27

Collective MPI

In some cases, such explicit control is not required and a commonly used communication pattern is sufficient. Suppose processor 0 has an expression in b that all processors are meant to have. A broadcast MPI call would do:

```
In[8]:=mpiBcast[b, 0, mpiCommWorld]
```

The second argument specifies which processor is the “root” of this broadcast; all others have their b overwritten. To collect values from all processors, use mpiGatherD:

```
In[9]:=mpiGather[b, c, 0, mpiCommWorld]
```

The variable c of processor 0 is written with a list of all the b of all the processors in mpiCommWorld. The temporal opposite is mpiScatter:

```
In[10]:=Clear[b]; a={2, 4, 5, 6}; mpiScatter[a, b, 0, mpiCommWorld]; b
```

```
Out[10]:={2, 4}
```

The mpiScatter command cuts up the variable a into even pieces (when possible) and scatters them to the processors. This is the result if \$NProc=2, but if \$NProc=4, b would only have {2}.

MPI provides reduction operations to perform simple computations mixed with messaging. Consider the following:

```
In[11]:=a={2+$IdProc, 45[ ], 3, {1+$IdProc, $NProc[ ]]};  
mpiReduce[a,d,mpiSum, 0,mpiCommWorld]
```

```
In[12]:=d
```

```
Out[12]:={{5, 90}, 6, {3, 4}}
```

The mpiSum constant indicates that variable a of every processor will be summed. In this case, \$NProc is 2, so those elements that were not identical result in odd sums, while those that were the same are even.

Most of these calls have default values if not all are specified. For example each of the following calls will have the equivalent effect as the above mpiGather[] call:

```
mpiGather[b, c, 0]
```

```
mpiGather[b, c]
```

```
c=mpiGather[b]
```

High-Level Calls

High-level calls can include convenient parallel versions of commonly used application program calls (for example, Mathematica calls). For example, ParallelTable[] is like Table[], except that the evaluations are automatically performed in a distributed manner:

```
In[13]:=ParallelTable[i,{i,100},0]
```

```
Out[13]:={1,2,3,4,5,...,99,100}
```

The third argument specifies that the answers are collated back to processor 0. This is a useful, simple way to parallelize many calls to a complex function. One could define a complicated function and evaluate it over a large range of inputs:

```
In[14]:=g[x_-]:=Gamma[2+0.5*(x-1)]; ParallelTable[g[i],  
{i,100},0]
```

```
Out[14]:={1, 1.32934, 2., 3.32335, 6., 11.6317, 24.,  
52.3428, 120., 287.885, 720}
```

ParallelFunctionToList[] also provides a simplified way to perform this form of parallelism.

Operations with Non-Trivial Communication

Matrix Operations

In some embodiments, one or more functions can help solve matrix calculations in parallel:

```
In[15]:=a=Table[i+3*$IdProc+2 j, {i, 2}, {j,4}]
```

```
Out[15]:={{3, 5, 7, 9}, {4, 6, 8, 10}}
```

```
In[16]:=t=ParallelTranspose[a]
```

```
Out[16]:{{3, 4, 6, 7}, {5, 6, 8, 9}}
```

Fourier Transforms

A Fourier transform of a large array can be solved faster in parallel, or made possible on a cluster because it can all be held in memory. A two-dimensional Fourier transform of the above example follows:

28

```
In[17]:=f=ParallelFourier[a]
```

```
Out[17]:={{32.+0. I, -4., -4. I, -4., -4.+4. I}, {-3.-3. I, 0.  
+0. I, 0., 0.+0. I}}
```

Edge Cell Management

Many problems require interactions between partitions, but only on the edge elements. Maintaining these edges can be performed using EdgeCell[]:

```
In[18]:=a={2, 4, 5, 6, 7}+8*$IdProc
```

```
Out[18]:={2, 4, 5, 6, 7}
```

```
10 In[19]:=EdgeCell[a]; a
```

```
Out[19]:={14, 4, 5, 6, 12}
```

Element Management

In particle-based problems, items can drift through space, sometimes outside the partition of a particular processor. This can be solved with ElementManage[1]:

```
In[20]:=list={{0,4},{1,3},{1,4},{0,5}}; fcn[x_]:=x[[1]]
```

```
In[21]:=ElementManage[list, fcn]
```

```
Out[21]:={{0, 4}, {0, 5}, {0, 4}, {0, 5}}
```

```
20 In[21]:=ElementManage[list, 2]
```

```
Out[21]:={{0, 4}, {0, 5}, {0, 4}, {0, 5}}
```

The second argument of ElementManage describes how to test elements of a list. The fcn identifier returns which processor is the “home” of that element. Passing an integer assumes that each element is itself a list, whose first element is a number ranging from 0 to the passed argument.

While the examples above involve Mathematica software and specific embodiments of MPI calls and cluster commands, it is recognized that these embodiments are used only to illustrate features of various embodiments of the systems and methods.

VI. Additional Embodiments

Although cluster computing techniques, modules, calls, and functions are disclosed with reference to certain embodiments, the disclosure is not intended to be limited thereby. Rather, a skilled artisan will recognize from the disclosure herein a wide number of alternatives for the exact selection of cluster calls, functions, and management systems. For example, single-node kernels can be managed using a variety of management tools and/or can be managed manually by a user, as described herein. As another example, a cluster node module can contain additional calls and procedures, including calls and procedures unrelated to cluster computing, that are not disclosed herein.

Other embodiments will be apparent to those of ordinary skill in the art from the disclosure herein. Moreover, the described embodiments have been presented by way of example only, and are not intended to limit the scope of the disclosure. Indeed, the novel methods and systems described herein can be embodied in a variety of other forms without departing from the spirit thereof. Accordingly, other combinations, omissions, substitutions and modifications will be apparent to the skilled artisan in view of the disclosure herein. Thus, the present disclosure is not intended to be limited by the disclosed embodiments, but is to be defined by reference to the appended claims. The accompanying claims and their equivalents are intended to cover forms or modifications as would fall within the scope and spirit of the inventions.

60 What is claimed is:

1. A computer cluster comprising:

a first processor;

a second processor;

a third processor;

at least one computer-readable medium in communication

at least one of the first processor, the second processor, or the third processor;

US 8,082,289 B2

29

a first kernel residing in the at least one computer-readable medium, said first kernel configured to translate commands into code for execution on the first processor;

a first cluster node module residing in the at least one computer-readable medium, said first cluster node module configured to send commands to the first kernel and receives commands from a user interface;

a second kernel residing in the at least one computer-readable medium, said second kernel configured to translate commands into code for execution on the second processor;

a second cluster node module residing in the at least one computer-readable medium, said second cluster node module configured to send commands to the second kernel and communicates with the first cluster node module;

a third kernel residing in the at least one computer-readable medium, said third kernel configured to translate commands into code for execution on the third processor; and

a third cluster node module residing in the at least one computer-readable medium, said third cluster node module configured to send commands to the third kernel and configured to communicate with the first cluster node module and the second cluster node module;

wherein the first cluster node module comprises a data structure in which messages originating from the second and third cluster node modules are stored.

2. The computer cluster of claim 1, wherein each of the first kernel, the second kernel, and the third kernel comprises a Mathematica kernel.

3. The computer cluster of claim 1, wherein each of the first cluster node module, the second cluster node module, and the third cluster node module comprises program code for implementing automatic load balancing on the first processor, the second processor, and the third processor.

4. The computer cluster of claim 1, further comprising a communications network for connecting at least two of the first processor, the second processor, or the third processor to one another.

5. The computer cluster of claim 1, wherein the at least one computer-readable medium comprises a first computer-readable medium coupled to at least the first processor and a second computer-readable medium coupled to at least the second processor, wherein the first kernel and the first cluster node module reside in the first computer-readable medium, and wherein the second kernel and the second cluster node module reside in the second computer-readable medium.

6. The computer cluster of claim 1, further comprising a cluster configuration module that initializes at least one of the first cluster node module, the second cluster node module, or the third cluster node module.

7. The computer cluster of claim 1, wherein the second cluster node module communicates with the first cluster node module using commands that implement at least a portion of a Message-Passing Interface.

8. The computer cluster of claim 1, wherein the data structure of the first cluster node module comprises a received message queue.

9. The computer cluster of claim 8, wherein the first cluster node module further comprises a message receiving queue, in which data specifying the location to which the first cluster node module can expect to receive an expression and an identifier of a processor from which the expression is expected to be sent are stored.

10. The computer cluster of claim 1, wherein the second cluster node module further comprises a received message

30

queue in which messages originating from the first and third cluster node modules are stored.

11. The computer cluster of claim 1, wherein the first processor and the second processor reside in the same computer system.

12. The computer cluster of claim 1, wherein the first processor and the second processor reside on the same die.

13. The computer cluster of claim 1, wherein the first cluster node module comprises an advanced functions module, and wherein the advanced functions module comprises a call that evaluates an expression across the computer cluster in parallel.

14. The computer cluster of claim 13, wherein the advanced functions module comprises a call that calculates a Fourier transform across the computer cluster in parallel.

15. The computer cluster of claim 13, wherein the advanced functions module comprises one or more calls that implement guard-cell operations.

16. The computer cluster of claim 13, wherein the advanced functions module comprises one or more calls that evaluate matrix operations across the computer cluster in parallel.

17. A computer cluster comprising:

a plurality of nodes, wherein each node is configured to access a computer-readable medium comprising program code for a user interface and program code for a single-node kernel module configured to interpret user instructions;

a plurality of cluster node modules, wherein each cluster node module is configured to communicate with a single-node kernel and with one or more other cluster node modules, to accept instructions from the user interface, and to interpret at least some of the user instructions such that the plurality of cluster node modules communicate with one another in order to act as a cluster; and

a communications network to connect the nodes; wherein one of the plurality of cluster node modules returns a result to the user interface.

18. The computer cluster of claim 17, wherein at least one of the plurality of cluster node modules comprises program code for storing data to one or more storage disks.

19. The computer cluster of claim 18, wherein the at least one of the plurality of cluster node modules further comprises program code for retrieving stored data from one or more storage disks.

20. The computer cluster of claim 17, wherein at least one of the plurality of cluster node modules comprises program code for collecting data from the plurality of cluster node modules and writing the data to a storage device associated with a single computer system.

21. The computer cluster of claim 17, wherein at least one of the plurality of cluster node modules comprises program code for reading data from a storage device associated with a single computer system and distributing the data to the plurality of cluster node modules within the computer cluster.

22. The computer cluster of claim 17, wherein at least one of the plurality of cluster node modules comprises program code for writing temporary data to a storage device associated with the same computer system on which the respective cluster node module is running.

23. The computer cluster of claim 17, wherein the communications network comprises at least one of a local bus, a local-area network, a wide-area network, an Ethernet network, a Token Ring network, a Frame Relay network, a Fiber Distributed Datalink Interface network, or an Asynchronous Transfer Mode network.

US 8,082,289 B2

31

24. The computer cluster of claim 17, wherein the user interface comprises a Mathematica front end.

25. The computer cluster of claim 17, wherein the user interface comprises a command line.

26. The computer cluster of claim 17, wherein each of the plurality of cluster node modules comprises a toolkit comprising library calls that implement at least a portion of a Message-Passing Interface.

27. The computer cluster of claim 17, wherein each of the plurality of cluster node modules comprises a toolkit comprising one or more high-level cluster computing commands.

28. The computer cluster of claim 17, wherein the cluster computer system comprises at least one of a plurality of Macintosh® computers, a plurality of Windows®-based computers, or a plurality of computers running a Unix or Linux operating system.

29. A method of evaluating a command on a computer cluster comprising:

communicating a command from at least one of a user interface or a script to one or more cluster node modules within the computer cluster;

for each of the one or more cluster node modules, communicating a message based on the command to a respective kernel module associated with the cluster node module;

for each of the one or more cluster node modules, receiving a result from the respective kernel module associated with the cluster node module; and

for at least one of the one or more cluster node modules, responding to messages from other cluster node modules.

30. The method of claim 29, wherein communicating a message based on the command to a respective kernel module associated with the cluster node module comprises communicating a specially identified message to the respective kernel module.

31. The method of claim 29, further comprising, for at least one of the cluster node modules, forwarding the result to at least one of a user interface or a script running on a same computer system as the cluster node module.

32. The method of claim 29, further comprising, for at least one of the cluster node modules, communicating the result in a message to one or more other cluster node modules.

33. The method of claim 29, wherein responding to messages from other cluster node modules comprises:

32

forwarding messages from the other cluster node modules to a received message queue;

matching data from each entry in a message receiving queue with entries in the received message queue;

combining data from the message receiving queue with matching data in the received message queue; and
marking the matching data as completed.

34. The method of claim 29, wherein communicating a command from at least one of a user interface or a script to one or more cluster node modules within the computer cluster comprises communicating an instruction from a Mathematica Front End to a first cluster node module, wherein the first cluster node module forwards the instruction to other cluster node modules running on the computer cluster.

35. The method of claim 34, wherein the first cluster node module forwards the instruction to other cluster node modules running on the computer cluster using a command from a Message-Passing Interface.

36. The method of claim 29, wherein each of the one or more cluster node modules communicates with a respective kernel module using MathLink.

37. A computing system for executing Mathematica code on multiple nodes comprising:

a first node module in communication with a first Mathematica kernel executing on a first node;

a second node module in communication with a second Mathematica kernel executing on a second node; and

a third node module in communication with a third Mathematica kernel executing on a third node;

wherein the first node module, the second node module, and the third node module are configured to communicate with one another using a peer-to-peer architecture.

38. The computing system of claim 37, wherein each of the first node module, the second node module, and the third node module comprises:

a data structure for maintaining messages originating from other node modules; and

a data structure for maintaining data specifying:

a location to which a message is expected to be received; and

an identifier for a node from which the message is expected to be sent.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 8,082,289 B2
APPLICATION NO. : 11/744461
DATED : December 20, 2011
INVENTOR(S) : Zvi Tannenbaum et al.

Page 1 of 1

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

On the Title Page:

In Item 56, page 2, column 2, line 24, under Other Publications, change “entitled” to --entitled--.

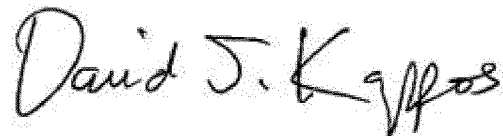
In the Specifications:

At column 6, line 46, change “13.0” to --130--.

At column 13, line 40 (Approx.), change “proessor” to --processor--.

At column 15, line 41, change “NPI” to --MPI--.

Signed and Sealed this
Sixth Day of November, 2012

A handwritten signature in black ink that reads "David J. Kappos". The signature is written in a cursive, flowing style with a large, stylized 'D' and 'K'.

David J. Kappos
Director of the United States Patent and Trademark Office

Exhibit 1



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
 United States Patent and Trademark Office
 Address: COMMISSIONER FOR PATENTS
 P.O. Box 1450
 Alexandria, Virginia 22313-1450
 www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
14/181,112	02/14/2014	Zvi Tannenbaum	ZTANN.002PIC3	6874

20995	7590	04/13/2018
KNOBBE MARTENS OLSON & BEAR LLP		
2040 MAIN STREET		
FOURTEENTH FLOOR		
IRVINE, CA 92614		

EXAMINER	
ASRES, HERMON	

ART UNIT	PAPER NUMBER
2449	

NOTIFICATION DATE	DELIVERY MODE
04/13/2018	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

jayna.cartee@knobbe.com
 efiling@knobbe.com

Office Action SummaryApplication No.
14/181,112Applicant(s)
TANNENBAUM ET AL.Examiner
HERMON ASRESArt Unit
2449AIA (First Inventor to File)
Status
No**-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --****Period for Reply**A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTHS FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 08/08/2017.
☐ A declaration(s)/affidavit(s) under **37 CFR 1.130(b)** was/were filed on ____.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ An election was made by the applicant in response to a restriction requirement set forth during the interview on ____; the restriction requirement and election have been incorporated into this action.
- 4) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims*

- 5) ☒ Claim(s) 2-41 is/are pending in the application.
 5a) Of the above claim(s) ____ is/are withdrawn from consideration.
- 6) ☐ Claim(s) ____ is/are allowed.
- 7) ☒ Claim(s) 2-41 is/are rejected.
- 8) ☐ Claim(s) ____ is/are objected to.
- 9) ☐ Claim(s) ____ are subject to restriction and/or election requirement.

* If any claims have been determined allowable, you may be eligible to benefit from the **Patent Prosecution Highway** program at a participating intellectual property office for the corresponding application. For more information, please see http://www.uspto.gov/patents/init_events/pph/index.jsp or send an inquiry to PPHfeedback@uspto.gov.

Application Papers

- 10) ☐ The specification is objected to by the Examiner.
- 11) ☒ The drawing(s) filed on 02/14/2014 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.
 Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
 Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

Certified copies:

- a) ☐ All b) ☐ Some** c) ☐ None of the:
- ☐ Certified copies of the priority documents have been received.
 - ☐ Certified copies of the priority documents have been received in Application No. ____.
 - ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

** See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- 1) ☒ Notice of References Cited (PTO-892)
- 2) ☒ Information Disclosure Statement(s) (PTO/SB/08a and/or PTO/SB/08b)
 Paper No(s)/Mail Date See Continuation Sheet.
- 3) ☐ Interview Summary (PTO-413)
 Paper No(s)/Mail Date. ____.
- 4) ☐ Other: ____.

Continuation Sheet (PTOL-326)

Application No. 14/181,112

Continuation of Attachment(s) 2). Information Disclosure Statement(s) (PTO/SB/08), Paper No(s)/Mail Date :08/08/2017, 09/22/2017, 12/01/2017.

Application/Control Number: 14/181,112
Art Unit: 2449

Page 2

DETAILED ACTION

Continued Examination Under 37 CFR 1.114

A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on August 08, 2017 has been entered.

Claims 2, 28, and 31 have been amended.

Claims 32-41 have been added.

Claims 2-41 are pending.

Response to Arguments

Double Patenting Rejection

The double patenting rejection will not be withdrawn because applicant in remarks filed on 08/08/2017 indicated that the double patenting rejection be held in abeyance until the other rejections are overcome.

Application/Control Number: 14/181,112
Art Unit: 2449

Page 3

35 U.S.C. 103 Rejections

Applicant's argument filed 08/08/2017 has been fully considered but they are deemed not persuasive in view of new grounds of rejection.

Double Patenting

The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. A nonstatutory double patenting rejection is appropriate where the claims at issue are not identical, but at least one examined application claim is not patentably distinct from the reference claim(s) because the examined application claim is either anticipated by, or would have been obvious over, the reference claim(s). See, e.g., *In re Berg*, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the reference application or patent either is shown to

Application/Control Number: 14/181,112
 Art Unit: 2449

Page 4

be commonly owned with this application, or claims an invention made as a result of activities undertaken within the scope of a joint research agreement. A terminal disclaimer must be signed in compliance with 37 CFR 1.321(b).

The USPTO internet Web site contains terminal disclaimer forms which may be used. Please visit <http://www.uspto.gov/forms/>. The filing date of the application will determine what form should be used. A web-based eTerminal Disclaimer may be filled out completely online using web-screens. An eTerminal Disclaimer that meets all requirements is auto-processed and approved immediately upon submission. For more information about eTerminal Disclaimers, refer to <http://www.uspto.gov/patents/process/file/efs/guidance/eTD-info-I.jsp>.

Claim 2, 28, and 31 are rejected on the ground of nonstatutory double patenting as being unpatentable over claims 1 and 8 of U.S. Patent No. 8,676,877 in view of Howard et al. (U.S. PGPub 2003/0195938). Although the conflicting claims are not identical, they are not patentably distinct from each other because: see table below.

Instant Application		US Patent 8,676,877	
Claim	Limitation	Claim	Limitation
2, 28, and 31	A computer cluster comprising:	1 and 8	A system for performing an instruction received from a front end by executing commands on one or more special purpose microprocessors, the system comprising:

Application/Control Number: 14/181,112
 Art Unit: 2449

Page 5

<p>a plurality of nodes, comprising a hardware processor wherein one or more of the nodes are configured to receive a command to start a cluster initialization process for the computer cluster, and</p> <p>wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that when executed causes the hardware processor to interpret user instructions <u>to evaluate mathematical expressions and to produce results of mathematical expression evaluation</u> ; and a mechanism for the nodes to communicate <u>results of mathematical expression evaluation</u> with each other using a peer-to-peer architecture; wherein at least one of the nodes is configured <u>to return at least one result of mathematical expression evaluation</u> to a user interface or a script</p>	<p>a plurality of nodes, wherein each node is configured to access a computer-readable memory system comprising program code for a single-node kernel module,</p> <p>wherein each cluster node module is stored in a computer-readable memory system and configured to communicate with a single-node kernel and with one or more other cluster node modules, to accept instructions, and to interpret at least some of the instructions such that the plurality of cluster node modules communicate with one another in order to act as a cluster in executing commands using one or more hardware processors</p>
--	--

It is noted that (U.S. Patent No. 8,676,877) teaches substantially the same limitation as Instant application's independent claims while failing to teach the limitation underlined in the table above. However Howard teaches this in paragraph [0017] "...transmitting an algorithm computation request and associated data from a requesting host to a home node of a computing system wherein the request includes a requested number (N) of processing nodes to be applied to computation of the request. ...distributes the computation request (Note: this is the mathematical expression) from the home node to a plurality of processing nodes wherein the plurality of processing

Application/Control Number: 14/181,112
Art Unit: 2449

Page 6

nodes includes N processing nodes coupled to the home node and wherein the distribution is in a hierarchical ordering. Then the method broadcasts the associated data from the home node to all of the plurality of processing nodes”. Howard in paragraph [0019] also teaches “...agglomerates a final computation result from partial computation results received from the plurality of processing nodes wherein the agglomeration is performed in the reverse order of the hierarchical ordering, and returns the final computation result from the home node to the requesting host”. Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine these teaching because this enables each remote host to communicate and parallel process algorithm code and data within the cluster but without direct communication with individual cluster nodes and/or detailed knowledge of cluster topology.

Claim Rejections - 35 USC § 103

The following is a quotation of pre-AIA 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

Claims 2, and 4-41 are rejected under 35 U.S.C. 103(a) as being unpatentable over Block et al. (U.S. PGPub 2005/0021751) hereinafter Block in

Application/Control Number: 14/181,112
Art Unit: 2449

Page 7

view of Singh (US Patent 8,601,101) hereinafter Singh and further in view of Howard et al. (U.S. PGPub 2003/0195938) hereinafter Howard.

As per claim 2, Block teaches a computer cluster ***(Block,cluster data port services within a cluster infrastructure. See paragraph [0025])*** comprising:

a plurality of nodes, comprising a hardware processor wherein one or more of the nodes are configured to receive a command to start a cluster initialization process for the computer cluster, and wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel ***(Block,multiple logical or TCP connections to a designated target node with designated backup target nodes, which appears to the user (i.e., an operating system or kernel component) like a single data port or data pipe. See paragraph [0044])*** that when executed causes the hardware processor to interpret user instructions; and a mechanism for the nodes to communicate with each other. ***(Block, establishment of multiple network connections between a source node, a target node and one or more backup nodes in such a manner that a cluster data port is effectively utilized as single data port. See paragraph [0025]).***

Block doesn't explicitly teach that the nodes communicate with each other using a peer-to-peer architecture and also that one of the nodes returns a result to a user interface or a script. In analogous art Singh teaches nodes that communicate with each other using a peer-to-peer architecture. ***(Singh,each node in a cluster including a peer-to-peer communication channel compiling and maintaining its own cluster***

Application/Control Number: 14/181,112
 Art Unit: 2449

Page 8

membership are disclosed..... each cluster node may be coupled to every other node and commonly-accessible storage through a network.Based on the cluster membership information, the joining node may request a peer-to-peer connection with each potential cluster member. See Column 2 line 26-38). Singh also teaches one of the nodes returns a result to a user interface or a script ***(Singh, The cluster configuration service 1510A may include a user interface to allow a system administrator to perform various node management functions through administrative console 1500. When the administrator makes changes to the node 240A configuration, the cluster configuration service 1510A may record all of the modifications that are made. See Column 14 line 61-66).***

Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to take the teaching of Singh and apply them on the teaching of Block because when each node on the cluster communication channel is in peer-to-peer connection, each node can detect the connection failure based on the socket communication with the failed node and can update its cluster membership data without the need for a central entity to pass an updated membership list among nodes. ***(Singh, See column 13 line 23-28).***

Block-Singh doesn't explicitly teach that the clusters evaluate mathematical expressions and produce results of Mathematica expressions. Block-Singh also doesn't teach return results of mathematical expression evaluation to a user interface. In analogous art Howard teaches clusters evaluate mathematical expressions and produce results of Mathematica expressions ***(Howard, ...transmitting an algorithm***

Application/Control Number: 14/181,112
 Art Unit: 2449

Page 9

computation request and associated data from a requesting host to a home node of a computing system wherein the request includes a requested number (N) of processing nodes to be applied to computation of the request. ...distributes the computation request (Note: this is the mathematical expression) from the home node to a plurality of processing nodes wherein the plurality of processing nodes includes N processing nodes coupled to the home node and wherein the distribution is in a hierarchical ordering. Then the method broadcasts the associated data from the home node to all of the plurality of processing nodes. See paragraph [0017]). Howard also teaches return results of mathematical expression evaluation to a user interface (*Howard, ...agglomerates a final computation result from partial computation results received from the plurality of processing nodes wherein the agglomeration is performed in the reverse order of the hierarchical ordering, and returns the final computation result from the home node to the requesting host. See paragraph [0017]. ...returning the partial result to the requesting host as a final result in response to further direction to complete processing of the complex algorithm. See paragraph [0019]).*

Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to take the teaching of Howard and apply them on the teaching of Block-Singh as doing so would enable a cluster of nodes to act as a single machine to one or more remote host computers outside of the inter-cluster network and also enables each remote host to communicate and parallel process algorithm code and

Application/Control Number: 14/181,112
Art Unit: 2449

Page 10

data within the cluster but without direct communication with individual cluster nodes and/or detailed knowledge of cluster topology. (**Howard, see paragraph [0068]**).

As per claim 4, the modified Block-Singh-Howard teaches wherein the mechanism comprises a message passing interface. (**Block, a single data port instance may also support target to source message sends. See paragraph [0050]**).

As per claim 5, the modified Block-Singh-Howard teaches wherein each of the nodes comprises one or more cluster node modules. (**Block, cluster data port services within a cluster infrastructure to provide reliable and efficient communications between nodes. See paragraph [0010]**).

As per claim 6, the modified Block-Singh-Howard teaches wherein each single-node kernel is stored in a non-transitory computer-readable medium and configured to accept and execute a request. (**Block, The routines executed to implementoperating system or a specific application, component, program, object, module or sequence of instructions, will also be referred to herein as "computer program code," or simply "program code." See paragraph [0039], [0049]**).

As per claim 7, the modified Block-Singh-Howard teaches wherein each of the nodes comprises one or more cluster node modules, wherein each of the cluster node

Application/Control Number: 14/181,112

Page 11

Art Unit: 2449

modules comprises instructions stored in a non-transitory computer- readable medium and wherein the instructions when executed by the hardware processor cause the cluster node module to communicate with the single-node kernel and with one or more other cluster node modules. ***(Block,multiple logical or TCP connections to a designated target node with designated backup target nodes, which appears to the user (i.e., an operating system or kernel component) like a single data port or data pipe. See paragraph [0044], [0039]).***

As per claim 8, the modified Block-Singh-Howard teaches wherein the plurality of cluster node modules act as a cluster. ***(Block, utilize cluster data port services within a cluster infrastructure to provide reliable and efficient communications between nodes in a clustered computer system. See paragraph [0010]).***

As per claim 9, the modified Block-Singh-Howard teaches wherein the plurality of cluster node modules communicate with one another to act as a cluster. ***(Block, Nodes 12, 14 and 16 are typically coupled together via a clustering network 18. See paragraph [0028]).***

As per claim 10, the modified Block-Singh-Howard teaches wherein the computer cluster includes the user interface. ***(Block, Cluster manager application 60 that provides the user interface whereby a user such as a systems administrator can manage clustering operations in the system. See paragraph [0037]).***

Application/Control Number: 14/181,112
Art Unit: 2449

Page 12

As per claim 11, the modified Block-Singh-Howard teaches wherein each cluster node module accepts instructions from the user interface and interprets one or more of the instructions. ***(Block, Cluster manager application 60 that provides the user interface whereby a user such as a systems administrator can manage clustering operations in the system. See paragraph [0037], [0039]).***

As per claim 12, the modified Block-Singh-Howard teaches wherein the cluster initialization process comprises establishing communication among two or more of the nodes. ***(Block, communications between nodes in a clustered computer system. See paragraph [0010]).***

As per claim 13, the modified Block-Singh-Howard teaches wherein the cluster initialization process comprises configuring access by one or more of the nodes to a computer-readable medium comprising program code for the single-node kernel. ***(Block, The cluster data port services described hereinafter, on the other hand, provide an abstracted transport service that encapsulates and manages the establishment of multiple logical or TCP connections to a designated target node with designated backup target nodes, which appears to the user (i.e., an operating system or kernel component) like a single data port or data pipe. See paragraph [0044]).***

Application/Control Number: 14/181,112
Art Unit: 2449

Page 13

As per claim 14, the modified Block-Singh-Howard teaches wherein the cluster initialization process comprises establishing message-passing support among the nodes in the cluster. ***(Block, data port services may also be configured to provide synchronous and asynchronous caller send models, as well as support message encryption. As such, the herein-described services may be used to provide a general messaging service that allows a variety of operating system or kernel components to make use of the services in a clustered environment. See paragraph [0045]).***

As per claim 15, the modified Block-Singh-Howard teaches wherein the cluster initialization process comprises launching cluster node modules by the cluster. ***(Block, see paragraph [0011], [0012]).***

As per claim 16, the modified Block-Singh-Howard teaches wherein the cluster initialization process comprises assigning a processor identification number to each of the cluster node modules. ***(Block, Data port internal object 72 also has a data port address table 78, which stores remote IP address, node ID pairs relating to the available IP addresses through which a particular node (identified by the node ID) is coupled to the clustering network. See paragraph [0059]).***

As per claim 17, the modified Block-Singh-Howard teaches wherein the cluster initialization process comprises: launching cluster node modules by the cluster; and

Application/Control Number: 14/181,112
Art Unit: 2449

Page 14

after launching the cluster node modules, configuring access by one or more of the nodes to a non-transitory computer-readable medium comprising program code for the single-node kernel. ***(Block, data port services may also be configured to provide synchronous and asynchronous caller send models, as well as support message encryption. As such, the herein-described services may be used to provide a general messaging service that allows a variety of operating system or kernel components to make use of the services in a clustered environment. See paragraph [0045]).***

As per claim 18, the modified Block-Singh-Howard teaches wherein the cluster initialization process comprises: launching cluster node modules by the cluster; after launching the cluster node modules, establishing communication among two or more of the nodes; ***(Block, a cluster data port consistent with the invention may be used to facilitate the transfer of large volumes of data between a source node and specified target nodes in a clustering environment. See paragraph [0012])*** and after establishing communication among two or more of the nodes, assigning a processor identification number to each of the cluster node modules. ***(Block, Data port internal object 72 also has a data port address table 78, which stores remote IP address, node ID pairs relating to the available IP addresses through which a particular node (identified by the node ID) is coupled to the clustering network. See paragraph [0059]).***

Application/Control Number: 14/181,112
Art Unit: 2449

Page 15

As per claim 19, the modified Block-Singh-Howard teaches wherein one or more of the nodes are configured to communicate at least some of the user instructions.

(Block, establishment of multiple network connections between a source node, a target node and one or more backup nodes in such a manner that a cluster data port is effectively utilized as single data port. See paragraph [0025], [0044]).

As per claim 20, the modified Block-Singh-Howard teaches wherein one or more of the nodes are configured to communicate at least some of the user instructions to one or more single-node kernel. ***(Block,multiple logical or TCP connections to a designated target node with designated backup target nodes, which appears to the user (i.e., an operating system or kernel component) like a single data port or data pipe. See paragraph [0044]).***

As per claim 21, the modified Block-Singh-Howard teaches wherein one or more of the nodes are configured to accept user instructions via one or more of the nodes. ***(Block, see paragraph [0025], [0044]).***

As per claim 22, the modified Block-Singh-Howard teaches wherein one or more of the nodes are configured to communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other. ***(Block, see paragraph [0025], [0044]).***

Application/Control Number: 14/181,112
Art Unit: 2449

Page 16

As per claim 23, the modified Block-Singh-Howard teaches wherein one or more of the nodes are configured to transmit at least some of the user instructions that originate from a user interface. ***(Singh, The cluster configuration service 1510A may include a user interface to allow a system administrator to perform various node management functions through administrative console 1500. When the administrator makes changes to the node 240A configuration, the cluster configuration service 1510A may record all of the modifications that are made. See Column 14 line 61-66).***

As per claim 24, the modified Block-Singh-Howard teaches wherein one or more of the nodes are configured to parallelize at least some of the user instructions before communicating at least some of the user instructions to one or more single-node kernels. ***(Block,multiple logical or TCP connections to a designated target node with designated backup target nodes, which appears to the user (i.e., an operating system or kernel component) like a single data port or data pipe. See paragraph [0044]).***

As per claim 25, the modified Block-Singh-Howard teaches wherein one or more of the nodes are configured to accept user instructions before communicating at least some of the user instructions to one or more single-node kernels. ***(Singh, The cluster configuration service 1510A may include a user interface to allow a system administrator to perform various node management functions through***

Application/Control Number: 14/181,112
Art Unit: 2449

Page 17

administrative console 1500. When the administrator makes changes to the node 240A configuration, the cluster configuration service 1510A may record all of the modifications that are made. See Column 14 line 61-66).

As per claim 26, the modified Block-Singh-Howard teaches wherein one or more of the nodes are configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; ***(Block, see paragraph [0025], [0044])*** and after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to one or more single-node kernels. ***(Block,multiple logical or TCP connections to a designated target node with designated backup target nodes, which appears to the user (i.e., an operating system or kernel component) like a single data port or data pipe. See paragraph [0044]). (Also see Singh Column 14 line 61-66).***

As per claim 27, the modified Block-Singh-Howard teaches wherein the plurality of nodes are configured to communicate with one another to interpret and translate commands for execution by a plurality of single-node kernels. ***(Block,multiple logical or TCP connections to a designated target node with designated backup target nodes, which appears to the user (i.e., an operating system or kernel component) like a single data port or data pipe. See paragraph [0044]). (Also see Singh Column 14 line 61-66).***

Application/Control Number: 14/181,112
Art Unit: 2449

Page 18

As per claim 28, Block teaches a computer cluster comprising: a plurality of nodes, **(Block,cluster data port services within a cluster infrastructure. See paragraph [0025])** wherein one or more of the nodes are configured to receive:

a command to start a cluster initialization process for the computer cluster, wherein the cluster initialization process comprises establishing communication among two or more of the nodes; **(Block, establishment of multiple network connections between a source node, a target node and one or more backup nodes in such a manner that a cluster data port is effectively utilized as single data port. See paragraph [0025]).**

and a mechanism for the nodes to communicate with each other using asynchronous calls; wherein each of the nodes is configured to access a non-transitory computer- readable medium comprising program code for a single-node kernel configured to interpret user instructions; and wherein at least one of the nodes is configured to return a result to the user interface or the script. **(Block,multiple logical or TCP connections to a designated target node with designated backup target nodes, which appears to the user (i.e., an operating system or kernel component) like a single data port or data pipe. See paragraph [0044]).**

Block doesn't explicitly teach receiving an instruction from a user interface or a script. In analogous art Singh teaches receiving instruction from a user interface in a computer cluster. **(Singh, The cluster configuration service 1510A may include a user interface to allow a system administrator to perform various node management functions through administrative console 1500. When the**

Application/Control Number: 14/181,112
 Art Unit: 2449

Page 19

administrator makes changes to the node 240A configuration, the cluster configuration service 1510A may record all of the modifications that are made. See Column 14 line 61-66).

Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to take the teaching of Singh and apply them on the teaching of Block because it would allow a system administrator to perform various node management functions through administrative console. ***(Singh, ... allow a system administrator to perform various node management functions through administrative console 1500. When the administrator makes changes to the node 240A configuration, the cluster configuration service 1510A may record all of the modifications that are made. See column 14 line 61-66).***

Block-Singh doesn't explicitly teach that the clusters evaluate mathematical expressions and produce results of Mathematica expressions. Block-Singh also doesn't teach return results of mathematical expression evaluation to a user interface. In analogous art Howard teaches clusters evaluate mathematical expressions and produce results of Mathematica expressions ***(Howard, ...transmitting an algorithm computation request and associated data from a requesting host to a home node of a computing system wherein the request includes a requested number (N) of processing nodes to be applied to computation of the request. ...distributes the computation request (Note: this is the mathematical expression) from the home node to a plurality of processing nodes wherein the plurality of processing nodes includes N processing nodes coupled to the home node and wherein the***

Application/Control Number: 14/181,112
 Art Unit: 2449

Page 20

distribution is in a hierarchical ordering. Then the method broadcasts the associated data from the home node to all of the plurality of processing nodes.

See paragraph [0017]). Howard also teaches return results of mathematical expression evaluation to a user interface (Howard, ...agglomerates a final computation result from partial computation results received from the plurality of processing nodes wherein the agglomeration is performed in the reverse order of the hierarchical ordering, and returns the final computation result from the home node to the requesting host. See paragraph [0017]. ...returning the partial result to the requesting host as a final result in response to further direction to complete processing of the complex algorithm. See paragraph [0019]).

Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to take the teaching of Howard and apply them on the teaching of Block-Singh as doing so would enable a cluster of nodes to act as a single machine to one or more remote host computers outside of the inter-cluster network and also enables each remote host to communicate and parallel process algorithm code and data within the cluster but without direct communication with individual cluster nodes and/or detailed knowledge of cluster topology. ***(Howard, see paragraph [0068]).***

As per claim 29, the modified Block-Singh-Howard teaches wherein the asynchronous calls comprise a first command to create a first packet containing: an expression to be sent as payload; and a target node where the expression should be sent; ***(Block, provide synchronous and asynchronous caller send models, as well***

Application/Control Number: 14/181,112
Art Unit: 2449

Page 21

as support message encryption. As such, the herein-described services may be used to provide a general messaging service that allows a variety of operating system or kernel components to make use of the services in a clustered environment. See paragraph [0045]) wherein the first command is configured to be called from within a single-node kernel; wherein the single-node kernel is configured to send the first packet to a local cluster node module; and wherein the local cluster node module is configured to forward the expression to the target node ***(Block,multiple logical or TCP connections to a designated target node with designated backup target nodes, which appears to the user (i.e., an operating system or kernel component) like a single data port or data pipe. See paragraph [0044]). (Also see Singh Column 14 line 61-66).***

As per claim 30,

[Rejection rational for claim 29 is applicable].

As per claim 31,

[Rejection rational for claim 2 and 28 is applicable].

As per claim 32, the modified Block-Singh-Howard teaches the computer cluster of claim 5, wherein each of the plurality of nodes implements asynchronous calls that enable the single-node kernel to perform computation tasks while the cluster node modules are simultaneously communicating with one another. ***(Block, ...cluster data***

Application/Control Number: 14/181,112
Art Unit: 2449

Page 22

port services may also be configured to provide synchronous and asynchronous caller send models, as well as support message encryption. As such, the herein-described services may be used to provide a general messaging service that allows a variety of operating system or kernel components to make use of the services in a clustered environment. See paragraph [0045]).

As per claim 33, the modified Block-Singh-Howard teaches the computer cluster of claim 5, wherein intercommunication among the plurality of single-node kernels during thread execution is enabled by the plurality of cluster node modules, and wherein the computer cluster is configured to permit exchange of information between nodes during the course of a parallel computation ***(Block, The cluster data port services described hereinafter, on the other hand, provide an abstracted transport service that encapsulates and manages the establishment of multiple logical or TCP connections to a designated target node with designated backup target nodes, which appears to the user (i.e., an operating system or kernel component) like a single data port or data pipe. See paragraph [0044]).***

As per claim 34, the modified Block-Singh-Howard teaches the computer cluster of claim 5, wherein each of the single-node kernels are configured to call a send command involving creating a packet containing: an expression to be sent as payload; ***(Block,multiple logical or TCP connections to a designated target node with designated backup target nodes, which appears to the user (i.e., an operating***

Application/Control Number: 14/181,112
 Art Unit: 2449

Page 23

system or kernel component) like a single data port or data pipe. See paragraph [0044]) and where the expression should be sent; wherein, upon reception of the send command by a local cluster node module associated with the single-node kernel, the local cluster node module is configured to decode the packet and forward a payload to the cluster node module specified in the packet; **(Block, ...message M1 is transmitted between the sockets on the source and target nodes. Upon receipt of the message, the socket reports to the data port of the target node that the record is complete (i.e., the message has been received and stored in a record of an IOCP queue). See paragraph [0084])** wherein each of the single-node kernels is configured to call a receipt command involving creating a packet specifying which message to test for completion and to then wait for a reply expression to evaluate; wherein, upon reception of the receipt command by the local cluster node module associated with the single-node kernel, the local cluster node module is configured to decode the packet and use a message specifier to search for any matching expressions listed as completed in a received message queue; **(Block, The data port notifies the user on the target node that such data has been received via a callback function rcvddata(M1,SN), where SN is the sequence number of the message. The user then processes the received message, and initiates the communication of an acknowledgment to the source node via a sendack(SN,RC) call to the data port, with RC storing a return code appropriate for the message. See paragraph [0084])** wherein, upon determining that such a completed expression is found, the local cluster node module is configured to send the completed expression to a local single- node

Application/Control Number: 14/181,112
Art Unit: 2449

Page 24

kernel associated with the cluster node module in response to the receipt command, and wherein each of the single-node kernels is configured to receive the completed expression and to update variables used by the single-node kernel during evaluation of expressions ***(Block, Once the acknowledgment is received by the socket, the socket notifies the data port that N acknowledgments have been received (to account for the possibility that multiple acknowledgments may be received). In response to such notification, the data port issues a callback function to the user to free the buffer allocated for the message M1, and to forward the return code RC generated by the target node. See paragraph [0085]).***

As per claim 35, the modified Block-Singh doesn't explicitly teach the computer cluster of claim 2 wherein the plurality of nodes are configured to permit exchange of information between nodes during the course of parallel computation. In analogous art Howard teaches the computer cluster of claim 2 wherein the plurality of nodes are configured to permit exchange of information between nodes during the course of parallel computation ***(Howard, ...parallel computing by providing methods and systems for configuration of and operation of a Howard Cascade (HC). Algorithm computation requests are transmitted to a home node of a HC and distributed (cascaded) through nodes of the HC to expand the algorithm computation request to a desired number of processing nodes. See paragraph [0013]).***

Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to take the teaching of Howard and apply them on the teaching

Application/Control Number: 14/181,112
 Art Unit: 2449

Page 25

of Block-Singh as doing so would enable a cluster of nodes to act as a single machine to one or more remote host computers outside of the inter-cluster network and also enables each remote host to communicate and parallel process algorithm code and data within the cluster but without direct communication with individual cluster nodes and/or detailed knowledge of cluster topology. (**Howard, see paragraph [0068]**).

As per claim 36, the modified Block-Singh-Howard teaches the computer cluster of claim 2, wherein each of the plurality of nodes comprises instructions executable by the hardware processor and configured to implement asynchronous behavior, wherein the instructions comprise: a first instruction to asynchronously send a payload to another node; (**Block, The cluster data port services may rely on sockets asynchronous I/O completion ports (IOCPs) to minimize the number of tasks required to support the service, given the potential for multiple logical or TCP connections being established between nodes. The services may also provide the structure to support asynchronous sends, receive callbacks, and improve efficiency for load balancing sends across multiple TCP connections. Enhanced socket support coupled with cluster data port services may also insure that ordered message delivery is maintained in spite of load balanced messaging across multiple connections simultaneously. See paragraph [0051]**) a second instruction to asynchronously receive a payload from another node; and a third instruction to search for a payload matching a message specifier. (**Block, ...cluster data port services may also be configured to provide synchronous and**

Application/Control Number: 14/181,112
Art Unit: 2449

Page 26

asynchronous caller send models, as well as support message encryption. As such, the herein-described services may be used to provide a general messaging service that allows a variety of operating system or kernel components to make use of the services in a clustered environment. See paragraph [0045]).

As per claim 37,

[Rejection rational for claim 2 and 28 is applicable].

As per claim 38, the modified Block-Singh doesn't explicitly teach the computer cluster node of claim 37, wherein the one or more non-transitory memory devices comprise program code for performing a parallel fast Fourier transform, wherein the program code causes the hardware processor to evaluate a command to perform a Fourier transform on an array comprising a first data portion that is stored on the computer cluster node and a second data portion that is not stored on the computer cluster node. In analogous art Howard teaches the computer cluster node of claim 37, wherein the one or more non-transitory memory devices comprise program code for performing a parallel fast Fourier transform, wherein the program code causes the hardware processor to evaluate a command to perform a Fourier transform on an array comprising a first data portion that is stored on the computer cluster node and a second data portion that is not stored on the computer cluster node (***Howard, Decomposition of Parallel Processing Data for Two-Dimensional Convolution Using Fast Fourier Transforms, see paragraph [0437], a two-dimensional convolution using Fast***

Application/Control Number: 14/181,112
Art Unit: 2449

Page 27

Fourier Transforms may be partitioned on a HC. The partitioning can handle an arbitrary number of parallel processing nodes and arbitrarily large images see paragraph [0438], [0439]).

Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to take the teaching of Howard and apply them on the teaching of Block-Singh as doing so would enable a cluster of nodes to act as a single machine to one or more remote host computers outside of the inter-cluster network and also enables each remote host to communicate and parallel process algorithm code and data within the cluster but without direct communication with individual cluster nodes and/or detailed knowledge of cluster topology. ***(Howard, see paragraph [0068]).***

As per claim 39,

[Rejection rational for claim 35 is applicable].

As per claim 40,

[Rejection rational for claim 36 is applicable].

As per claim 41, the modified Block-Singh doesn't explicitly teach the computer cluster node of claim 37 wherein the hardware processor comprises a special purpose microprocessor. In analogous art Howard teaches the computer cluster node of claim 37 wherein the hardware processor comprises a special purpose microprocessor ***(Howard, ...a first processor 420 connected to a second processor 422 via a***

Application/Control Number: 14/181,112
Art Unit: 2449

Page 28

single communication channel 424. (Note: this are special purpose processors)
Multiple parallel channels of expansion, on the other hand, consist of multiple computer processors and multiple communication channels, as shown in FIG. 33. Processor 430 and processor 432 of processing node 112 are connected to processor 34 and processor 436 of processing node 114 by two communication channels 438 and 440, respectively. Other parallel channel configurations follow and may be implemented such that a HC moves data as efficiently as desired. See paragraph [0167]).

Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to take the teaching of Howard and apply them on the teaching of Block-Singh as doing so would enable a cluster of nodes to act as a single machine to one or more remote host computers outside of the inter-cluster network and also enables each remote host to communicate and parallel process algorithm code and data within the cluster but without direct communication with individual cluster nodes and/or detailed knowledge of cluster topology. ***(Howard, see paragraph [0068]).***

Claim 3 is rejected under 35 U.S.C. 103(a) as being unpatentable over Block et al. (US PGPub 2005/0021751) hereinafter Block in view of Singh (US Patent 8,601,101) hereinafter Singh and further in view of Gray (USPGPub 2007/0073705) and further in view of Howard et al. (U.S. PGPub 2003/0195938) herein Howard.

Application/Control Number: 14/181,112
Art Unit: 2449

Page 29

As per claim 3, Block-Singh-Howard teaches the computer cluster of claim 2 but doesn't teach a Mathematica kernel. In analogous art Gray teaches a Mathematica kernel. ***(Gray, The front end may include an interactive document referred to as a notebook similar to those often used with MATHEMATICA.RTM. software systems. A notebook may include input to be sent to the kernel 104 and output received from the kernel, as well as text, graphics, palettes, etc. see paragraph [0031]).***

Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to take the teaching of Gray and apply them on the teaching of Block-Singh-Howard because the kernel and the front end may be implemented on a same computing system or on different computing systems that are communicatively coupled to one another. ***(Gray, see paragraph [0030]).***

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to HERMON ASRES whose telephone number is (571)272-4257. The examiner can normally be reached on Monday - Friday 7:30 am to 5:00 pm est.

Examiner interviews are available via telephone, in-person, and video conferencing using a USPTO supplied web-based collaboration tool. To schedule an interview, applicant is encouraged to use the USPTO Automated Interview Request (AIR) at <http://www.uspto.gov/interviewpractice>.

Application/Control Number: 14/181,112
Art Unit: 2449

Page 30

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, VIVEK SRIVASTAVA can be reached on 571-272-7304. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/HERMON ASRES/
Examiner, Art Unit 2449

Exhibit 2

Trials@uspto.gov
571-272-7822

Paper 11
Date: May 5, 2021

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

NVIDIA CORP.,
Petitioner,

v.

ADVANCED CLUSTER SYSTEMS, INC.,
Patent Owner.

IPR2020-01608
Patent 8,082,289 B2

Before KARL D. EASTHOM, ARTHUR M. PESLAK, and
SEAN P. O'HANLON, *Administrative Patent Judges*.

O'HANLON, *Administrative Patent Judge*.

DECISION
Granting Institution of *Inter Partes* Review
35 U.S.C. § 314

IPR2020-01608
 Patent 8,082,289 B2

I. INTRODUCTION

A. Background

NVIDIA Corporation (“Petitioner”) filed a Petition for *inter partes* review of claims 1, 4–6, 8, 10, 11, 13, 14, 16–19, 21–23, 27, and 29–32 (“the challenged claims”) of U.S. Patent No. 8,082,289 B2 (Ex. 1001, “the ’289 patent”). Paper 1 (“Pet.”), 1. Advanced Cluster Systems, Inc. (“Patent Owner”) filed a Preliminary Response. Paper 7 (“Prelim. Resp.”). Pursuant to our order (Paper 8), Petitioner filed a Reply to Patent Owner’s Preliminary Response (Paper 9, “Pet. Reply”) and Patent Owner filed a Sur-reply to Petitioner’s Reply (Paper 10, “PO Sur-reply”).

Institution of an *inter partes* review is authorized by statute only when “the information presented in the petition . . . and any response . . . shows that there is a reasonable likelihood that the petitioner would prevail with respect to at least 1 of the claims challenged in the petition.” 35 U.S.C. § 314(a) (2018). A decision to institute may not institute on fewer than all claims challenged in the petition. *SAS Inst. Inc. v. Iancu*, 138 S. Ct. 1348, 1354, 1359–60 (2018). If the PTAB institutes a trial, the PTAB will institute on all challenges raised in the petition. *See* Patent Trial and Appeal Board Consolidated Trial Practice Guide 64 (Nov. 2019) (“The Board will not institute on fewer than all claims or all challenges in a petition.”);¹ *see also AC Techs. S.A. v. Amazon.com, Inc.*, 912 F.3d 1358, 1364 (Fed. Cir. 2019) (“[I]f the Board institutes an IPR, it must . . . address all grounds of unpatentability raised by the petitioner.”).

¹ Available at <https://www.uspto.gov/TrialPracticeGuideConsolidated>.

IPR2020-01608
Patent 8,082,289 B2

For the reasons set forth below, upon considering the parties' briefs and evidence of record, we conclude that the information presented shows that there is a reasonable likelihood that Petitioner would prevail in establishing the unpatentability of at least one of the challenged claims. Thus, we institute *inter partes* review of all challenged claims based on all asserted grounds.

B. Real Parties in Interest

Petitioner identifies itself as the sole real party in interest. Pet. 3.

Patent Owner identifies itself as the sole real party in interest.

Paper 3, 1.

C. Related Matters

The parties indicate that the '289 patent is the subject of the following district court proceeding:

Advanced Cluster Systems, Inc. v. NVIDIA Corp.,
No. 19-cv-02032 (D. Del. filed Oct. 28, 2019).

Pet. 3; Paper 3, 1. Patent Owner further notes various patents and patent applications in the priority chain of the '289 patent and petitions for *inter partes* review concerning three of the identified patents. Paper 3, 1; Paper 4, 1.

D. The Challenged Patent

The '289 patent is titled "Cluster Computing Support for Application Programs" and discloses "systems and methods for adding cluster computing functionality to a computer program." Ex. 1001, code (54), 1:17–20. The '289 patent recognizes that computer clusters "include a group of two or

IPR2020-01608
Patent 8,082,289 B2

more computers, microprocessors, and/or processor cores (‘nodes’) that intercommunicate so that the nodes can accomplish a task as though they were a single computer.” *Id.* at 1:22–25. Grid computing is one manner in which nodes can cooperatively act together. *Id.* at 1:49–51. One form of grid computing, known as “distributed computing,” involves a master node that manages a plurality of slave nodes or computational nodes, which work independently and receive commands and data only from the master node. *Id.* at 1:53–67. However, the nodes “generally do not communicate with one another as peers.” *Id.* at 1:52–53.

The ’289 patent purports to improve upon grid computing by “adding cluster computing functionality to a computer application.” Ex. 1001, 2:8–10. Figure 1 shows a block diagram of a computer cluster system and is reproduced below:

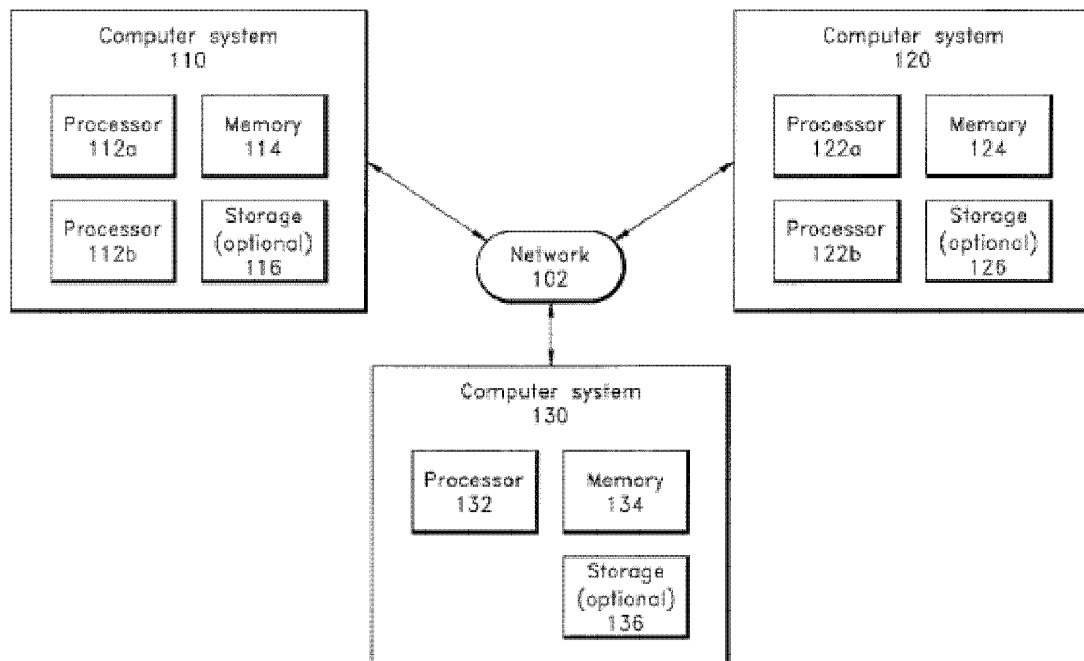


FIG. 1

IPR2020-01608
Patent 8,082,289 B2

Figure 1 is a block diagram of a computer cluster wherein computer systems 110, 120, 130 communicate with one another via communications network 102. *Id.* at 4:59–62. Each computer system includes at least one processor 112a, 112b, 122a, 122b, 132, memory 114, 124, 134, and, optionally, storage 116, 126, 136. *Id.* at 4:63–5:2. Each processor includes an independent processing core, or “node,” that is capable of single-threaded execution. *Id.* at 4:39–44, 5:2–7.

Figure 2 shows a block diagram of the computer cluster’s software modules and is reproduced below:

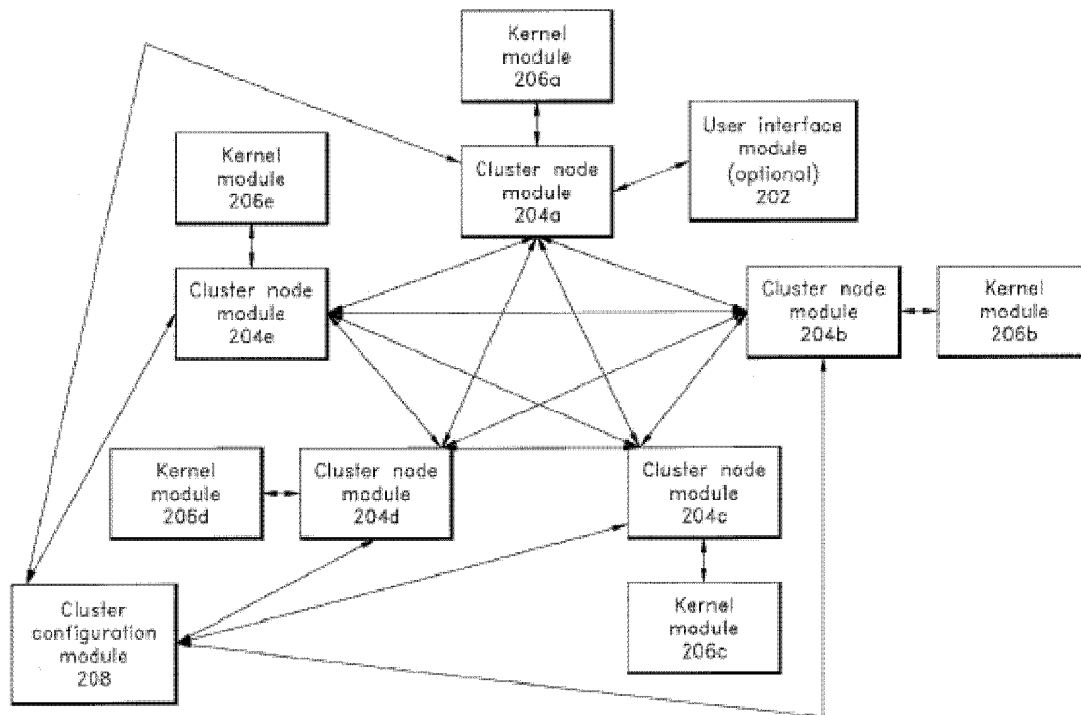


FIG. 2

Figure 2 is a block diagram showing the relationships among software modules running on one embodiment of computer cluster 100. Ex. 1001, 5:12–14. Software modules, or “kernels,” run on the nodes within the interconnected computer systems. *Id.* at 4:52–54, 23:1–17. A kernel

IPR2020-01608
Patent 8,082,289 B2

“executes instructions provided to the program by a user, a script, or another source” and “can manage at least some hardware resources of a computer system and/or can manage communications between those resources and software.” *Id.* at 1:33–41. The kernel modules are designed for single-threaded execution. *Id.* at 5:14–15. Software code designed for single-threaded execution can generally run on one node at a time. *Id.* at 5:7–9.

Each node includes a cluster node module in communication with a single kernel module. Ex. 1001, 5:29–31. Cluster node modules are software modules that include at least a portion of the message-passing interface (MPI) application programming interface (API) to interact with an application, such as Mathematica. *Id.* at 11:29–45. In addition to communicating with its respective kernel module in the embodiment of Figure 2, each cluster node module is also in communication with each of the other cluster node modules. *Id.* at 5:37–40. One of the cluster node modules (module 204a) is in communication with user interface module 202. *Id.* at 11:2–5. That cluster node module receives commands from the user interface and submits the commands to all of the other cluster node modules. *Id.* at 24:38–44. Each cluster node module communicates the command to its respective kernel module. *Id.* at 24:54–60. Each kernel module processes the command and returns a result to its respective cluster node module. *Id.* at 24:61–65. The cluster node module can report the result to the other cluster node modules. *Id.* at 24:65–25:1. This peer-to-peer behavior of the cluster node modules allows code running within multiple, simultaneously running kernel modules to interact on a collective basis, performing calculations, processing, or other work on a larger scale and faster than one kernel acting alone. *Id.* at 25:21–28.

IPR2020-01608
 Patent 8,082,289 B2

E. The Challenged Claims

Petitioner challenges claims 1, 4–6, 8, 10, 11, 13, 14, 16–19, 21–23, 27, and 29–32 of the '289 patent. Pet. 1. Claims 1, 17, and 29 are independent. Claim 1 is illustrative of the challenged claims and is reproduced below:

1. A computer cluster comprising:
 - a first processor;
 - a second processor;
 - a third processor;
 - at least one computer-readable medium in communication at least one of the first processor, the second processor, or the third processor;
 - a first kernel residing in the at least one computer-readable medium, said first kernel configured to translate commands into code for execution on the first processor;
 - a first cluster node module residing in the at least one computer-readable medium, said first cluster node module configured to send commands to the first kernel and receives commands from a user interface;
 - a second kernel residing in the at least one computer-readable medium, said second kernel configured to translate commands into code for execution on the second processor;
 - a second cluster node module residing in the at least one computer-readable medium, said second cluster node module configured to send commands to the second kernel and communicates with the first cluster node module;
 - a third kernel residing in the at least one computer-readable medium, said third kernel configured to translate commands into code for execution on the third processor; and
 - a third cluster node module residing in the at least one computer-readable medium, said third cluster node module configured to send commands to the third kernel and configured to communicate with the first cluster node module and the second cluster node module;

IPR2020-01608
Patent 8,082,289 B2

wherein the first cluster node module comprises a data structure in which messages originating from the second and third cluster node modules are stored.

Ex. 1001, 28:61–29:28.

F. Asserted Grounds of Unpatentability

The Petition relies on the following references in challenging the claims of the '289 patent:

Schreiner1: Wolfgang Schreiner et al., *Distributed Maple: Parallel Computer Algebra in Networked Environments*, 35 Journal of Symbolic Computation 305 (2003), filed as Exhibit 1008;

Schreiner2: Wolfgang Schreiner, *Distributed Maple – User and Reference Manual (V 1.1.12)* (2001), filed as Exhibit 1009;

Schreiner3: Károly Bósa and Wolfgang Schreiner, *Taks Logging, Rescheduling and Peer Checking in Distributed Maple* (2002), filed as Exhibit 1010;

Maple Guide: K. M. Heal et al., *Maple V Learning Guide* (J. S. Devitt ed., 1998), filed as Exhibit 1011;

Dist.Maple5: “Source code for parallel versions of Maple functions in Distributed Maple from the ‘distsoft’ directory” (Pet. xii), filed as Exhibit 1012;

CASA Function Source Code: “Source code for parallel versions of Maple functions in Distributed Maple from the ‘distsoft’ directory” (Pet. xii), filed as Exhibit 1013;

Maple Function Source Code: “Source code for parallel versions of CASA functions in Distributed Maple from the ‘distsoft’ directory” (Pet. xii), filed as Exhibit 1014;

Install1 File: “‘Install’ file for Distributed Maple” (Pet. xii), filed as Exhibit 1015;

ReadMe1 File: “‘ReadMe’ file for Distributed Maple” (Pet. xii), filed as Exhibit 1016;

IPR2020-01608
Patent 8,082,289 B2

Install2 File: “‘Install’ file for source code in ‘distsoft’ directory” (Pet. xii), filed as Exhibit 1017;

ReadMe2 File: “‘ReadMe’ file for source code in ‘distsoft’ directory” (Pet. xii), filed as Exhibit 1018;

Howard: US 2003/0195938 A1, published Oct. 16, 2003, filed as Exhibit 1019; and

Maple Reference: Michael Kofler, *Maple An Introduction and Reference* (1997), filed as Exhibit 1031.

Petitioner refers to Exhibits 1008–1010 and 1012–1018 collectively as “Distributed Maple Publications.” Pet. 7–8. Petitioner refers to Exhibits 1012–1018 collectively as “Distributed Maple Code.” *Id.* at 8.

Petitioner asserts the following grounds of unpatentability:

Claim(s) Challenged	35 U.S.C. §	References
1, 4–6, 8, 10, 11, 13, 16–19, 21–23, 27, 29–32	103(a) ²	Schreiner1, Schreiner2, Schreiner3, Maple Guide, Distributed Maple Code
14	103(a)	Schreiner1, Schreiner2, Schreiner3, Maple Guide, Distributed Maple Code, Maple Reference, Howard

Pet. 10. Petitioner submits declarations of Henry Tufo, Ph.D. (Ex. 1005, “Tufo Declaration”) and Wolfgang Schreiner, Ph.D. (Ex. 1006, “Schreiner Declaration”) in support of its contentions. Patent Owner submits declarations of Jaswinder Pal Singh, Ph.D. (Ex. 2001), Dean Dager, Ph.D.

² The application resulting in the ’289 patent was filed prior to the date when the Leahy-Smith America Invents Act (“AIA”), Pub. L. No. 112–29, 125 Stat. 284 (2011), took effect. Thus, we refer to the pre-AIA version of sections 103 and 112 herein.

IPR2020-01608
Patent 8,082,289 B2

(Ex. 2006, “Dauger Declaration”), Vineer Bhansali, Ph.D. (Ex. 2007), and John Bancroft (Ex. 2008) in support of its preliminary responses.

II. ANALYSIS

A. Principles of Law

Petitioner bears the burden of persuasion to prove unpatentability, by a preponderance of the evidence, of the claims challenged in the Petition. 35 U.S.C. § 316(e). This burden never shifts to Patent Owner. *Dynamic Drinkware, LLC v. Nat’l Graphics, Inc.*, 800 F.3d 1375, 1378 (Fed. Cir. 2015). The Board may authorize an *inter partes* review if we determine that the information presented in the Petition and Patent Owner’s Preliminary Response shows that there is a reasonable likelihood that Petitioner would prevail with respect to at least one of the claims challenged in the Petition. 35 U.S.C. § 314(a).

A patent claim is unpatentable under 35 U.S.C. § 103(a) if the differences between the claimed subject matter and the prior art are such that the subject matter, as a whole, would have been obvious at the time the invention was made to a person having ordinary skill in the art to which the subject matter pertains. *KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 406 (2007). The question of obviousness is resolved on the basis of underlying factual determinations including (1) the scope and content of the prior art, (2) any differences between the claimed subject matter and the prior art, (3) the level of skill in the art, and (4) when in evidence, any objective evidence of nonobviousness. *Graham v. John Deere Co.*, 383 U.S. 1, 17–18 (1966).

IPR2020-01608
Patent 8,082,289 B2

B. Level of Ordinary Skill in the Art

The level of ordinary skill in the art is “a prism or lens” through which we view the prior art and the claimed invention. *Okajima v. Bourdeau*, 261 F.3d 1350, 1355 (Fed. Cir. 2001). The person of ordinary skill in the art is a hypothetical person presumed to have known the relevant art at the time of the invention. *In re GPAC Inc.*, 57 F.3d 1573, 1579 (Fed. Cir. 1995). In determining the level of ordinary skill in the art, we may consider certain factors, including the “type of problems encountered in the art; prior art solutions to those problems; rapidity with which innovations are made; sophistication of the technology; and educational level of active workers in the field.” *Id.* (internal quotation marks and citation omitted).

Dr. Tufo testifies that a person having ordinary skill in the art at the time of the invention (“POSITA”) would have had “a Bachelor’s degree in computer science, electrical engineering, or an equivalent field, and two years of academic or industry experience in parallel and distributed computing.” Ex. 1005 ¶ 40; *see* Pet. 13 (citing Ex. 1005 ¶¶ 38–41).

Patent Owner does not proffer a definition for the level of ordinary skill in the art or refute that proposed by Petitioner. *See generally* Prelim. Resp.

For purposes of this Decision on Institution, we adopt Petitioner’s proposed level of ordinary skill in the art, which comports with the teachings of the ’289 patent and the asserted prior art.

C. Claim Construction

In an *inter partes* review, claims are construed using the same claim construction standard that would be used to construe the claims in a civil

IPR2020-01608
Patent 8,082,289 B2

action under 35 U.S.C. § 282(b), including construing the claims in accordance with the ordinary and customary meaning of such claims as understood by one of ordinary skill in the art and the prosecution history pertaining to the patent. 37 C.F.R. § 42.100(b) (2020). Thus, we apply the claim construction standard as set forth in *Phillips v. AWH Corp.*, 415 F.3d 1303 (Fed. Cir. 2005) (en banc). In addition to the specification and prosecution history, we also consider use of the terms in other claims and extrinsic evidence including expert and inventor testimony, dictionaries, and learned treatises, although extrinsic evidence is less significant than the intrinsic record. *Phillips*, 415 F.3d at 1312–17. Usually, the specification is dispositive, and it is the single best guide to the meaning of a disputed term. *Id.* at 1315.

The specification may reveal a special definition given to a claim term by the patentee, or the specification may reveal an intentional disclaimer or disavowal of claim scope by the inventor. *Phillips*, 415 F.3d at 1316. If an inventor acts as his or her own lexicographer, the definition must be set forth in the specification with reasonable clarity, deliberateness, and precision. *Renishaw PLC v. Marposs Societa' per Azioni*, 158 F.3d 1243, 1249 (Fed. Cir. 1998).

Disavowal of a claim term “can be effectuated by language in the specification or the prosecution history.” *Poly-America, L.P. v. API Indus., Inc.*, 839 F.3d 1131, 1136 (Fed. Cir. 2016). “In either case, the standard for disavowal is exacting, requiring clear and unequivocal evidence that the claimed invention includes or does not include a particular feature.” *Id.* (citing *Openwave Sys., Inc. v. Apple Inc.*, 808 F.3d 509, 513–14 (Fed. Cir. 2015); *Omega Eng'g, Inc. v. Raytek Corp.*, 334 F.3d 1314, 1323–26 (Fed.

IPR2020-01608
Patent 8,082,289 B2

Cir. 2003)). “Ambiguous language cannot support disavowal.” *Id.* (citing *Omega*, 334 F.3d at 1324).

Although disavowal must be clear and unequivocal, it need not be explicit. *Trs. of Columbia Univ. v. Symantec Corp.*, 811 F.3d 1359, 1363–64 (Fed. Cir. 2016). For example, an inventor may disavow claim scope lacking a particular feature when the specification describes “the present invention” as having that feature. *See, e.g., Luminara Worldwide, LLC v. Liown Elecs. Co.*, 814 F.3d 1343, 1353 (Fed. Cir. 2016). Similarly, an inventor may disavow claim scope lacking a particular feature when the specification distinguishes or disparages prior art based on the absence of that feature. *See Openwave*, 808 F.3d at 513–14; *SightSound Techs., LLC v. Apple Inc.*, 809 F.3d 1307, 1317 (Fed. Cir. 2015).

Only those terms that are in controversy need be construed, and only to the extent necessary to resolve the controversy. *Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co.*, 868 F.3d 1013, 1017 (Fed. Cir. 2017) (citing *Vivid Techs., Inc. v. Am. Sci. & Eng’g, Inc.*, 200 F.3d 795, 803 (Fed. Cir. 1999)).

Petitioner argues that “no express construction of any term is needed to resolve the challenges” in the Petition. Pet. 13.

Patent Owner presents two separate requirements and constructions for the “cluster node module” as recited in each of the challenged claims. First, Patent Owner argues that “[e]very challenged claim of the ’289 patent includes one or more limitations related to the relative order in which commands or instructions are processed by the user interface, cluster node modules, and kernels.” Prelim. Resp. 9. Patent Owner argues that “first cluster node module configured to send commands to the first kernel and

IPR2020-01608
 Patent 8,082,289 B2

receives commands from a user interface” as recited in claim 1 should be interpreted

as establishing the following relative order in which commands are processed by the user interface, first cluster node module, and first kernel: (1) first, commands start at the user interface, (2) second, commands are “receive[d] from” the user interface by the first cluster node module, and, (3) third, commands are “sen[t] . . . to” the first kernel by the first cluster node module.

Id. at 12 (alterations in original). According to Patent Owner, “[t]his excludes the first kernel receiving commands from the user interface and forwarding commands to the first cluster node module.” *Id.* (citing Ex. 2001 ¶ 45). Patent Owner argues that similar recitations in independent claims 17 and 29 should be interpreted in the same manner. *Id.* at 17, 20–21 (citing Ex. 2001 ¶¶ 52, 56).

Second, Patent Owner contends,

the Board should construe “cluster node module” to mean “a module that cooperates with other cluster node modules to establish intercommunication among nodes in a computer cluster and to exchange messages such that each node can communicate tasks and data with other nodes *without the tasks and data being required to go through a central server or master node.*”

Prelim. Resp. 22 (emphasis added) (quoting Ex. 2001 ¶ 58).

Regarding the first requirement, Patent Owner argues claim 1 “excludes the first kernel receiving commands from the user interface and forwarding commands to the first cluster node module.” Prelim. Resp. 12 (citing Ex. 2001 ¶ 45). Patent Owner sets forth its argument in the following diagram:

IPR2020-01608
Patent 8,082,289 B2

Correct order:	User Interface → Cluster Node Module → Kernel
Incorrect order:	User Interface → Kernel → Cluster Node Module

The diagram above represents Patent Owner’s claim construction, wherein construing claim 1 to allow a kernel to pass an instruction from a user interface to a cluster node module is labeled “incorrect.” *See id.* at 1–2 (citing Ex. 2001 ¶ 37).

Contrary to Patent Owner’s arguments, the plain language of claim 1 does not exclude instructions passing through an intervening kernel and does not require any of the negative limitations argued. Rather, claim 1 recites “said first cluster node module configured to send commands to the first kernel and receives commands from a user interface.” Ex. 1001, 29:5–7. This language requires “said first cluster node module . . . to send commands to the first kernel” without specifying the first kernel’s position relative to a user interface and cluster node module, and also without specifying anything about a central server or master mode. The language also requires “said first cluster node module . . . to . . . receive[] commands from a user interface,” which does not prevent the instructions from passing through a kernel, central server, or master node situated between the user interface and the first cluster node module.

Contrary to Patent Owner’s other arguments, the specification of the ’289 patent does not limit the claims in the manner argued. *See* Prelim. Resp. 10–12 (citing Ex. 1001, 22:60–62, Fig. 2). Patent Owner relies on Figure 2 and other selected passages to incorporate limitations into claim 1 (*see id.*), but the specification specifically states that “[t]he drawings and the associated descriptions are provided to illustrate embodiments and *not to*

IPR2020-01608
 Patent 8,082,289 B2

limit the scope of the disclosure.” Ex. 1001, 3:61–63 (emphasis added). In addition, “[w]hile we read claims in view of the specification, of which they are a part, we do not read limitations from the embodiments in the specification into the claims.” *Hill-Rom Servs., Inc. v. Stryker Corp.*, 755 F.3d 1367, 1371 (Fed. Cir. 2014) (citing *Liebel–Flarsheim Co. v. Medrad, Inc.*, 358 F.3d 898, 904 (Fed. Cir. 2004)).

Similarly, the specification describes several embodiments under a “SUMMARY” of the invention section in general terms “[w]ithout limiting the scope of the invention.” *See* Ex. 1001, 2:5–7. One passage mimics the broad language of claim 1, which plainly does not provide the negative limitations argued by Patent Owner: “The first cluster node module is configured to send commands to the first kernel and receives commands from a user interface.” *Id.* at 2:41–43. This generic passage allows the first cluster node module to “receive[] commands from a user interface” without requiring it to accept them *directly* from the user interface. The passage says nothing about precluding passage of a message through a master node, central server, or kernel.

Patent Owner also relies on Figure 2 as “show[ing] that the user interface module 202 is connected to the cluster node module 204a **only** and the kernel module 206a is connected to the cluster node module 204a **only**.” Prelim. Resp. 11. Patent Owner similarly contends that “[t]here is no connection between the user interface module 202 and the kernel module 206a or any of the other kernel modules.” *Id.*

The specification contradicts Patent Owner on this preliminary record. For example, it states: “A kernel module 206 typically includes program code for interpreting high-level code, commands, and/or instructions

IPR2020-01608
 Patent 8,082,289 B2

supplied by a user or a script into low-level code, such as, for example, machine language or assembly language.” Ex. 1001, 23:2–5 (emphasis added). In other words, contrary to Patent Owner’s arguments, on this preliminary record, the specification supports connecting a user interface directly to any kernel.

In line with the above teachings, the specification describes user interface module 202 communicating with kernel module 202 for “some embodiments” as follows:

In some embodiments, computer cluster 100 includes a user interface module 202, such as, for example a Mathematica Front End or a command line interface, that includes program code for a kernel module 206 to provide graphical output, accept graphical input, and provide other methods of user communication that a graphical user interface or a command-line interface provides. To support a user interface module 202, the behavior of a cluster node module 204a is altered in some embodiments. Rather than sending output to and accepting input from the user directly, the user interface module 202 activates the cluster node module 204a to which it is connected and specifies parameters to form a connection, such as a MathLink connection, between the cluster node module 204a and the user interface module 202. The user interface module’s activation of the cluster node module 204a can initiate the execution of instructions to activate the remaining cluster node modules 204b-e on the cluster and to complete the sequence to start all kernel modules 206a-e on the cluster. Packets from the user interface module 202, normally intended for a kernel module 206a, are accepted by the cluster node module 204a as a user command. Output from the kernel module 206a associated with the cluster node module 204a can be forwarded back to the user interface module 202 for display to a user. Any of the cluster node modules 204a-e can be configured to communicate with a user interface module 202.

Ex. 1001, 22:42–67 (emphases added).

IPR2020-01608
Patent 8,082,289 B2

Patent Owner argues that portions of this passage support its construction. Prelim. Resp. 10 (citing Ex. 1001, 22:60–62). However, in context, the passage, read in its entirety, does not support Patent Owner’s limited claim construction on this preliminary record. Rather, the first emphasized portion as quoted above explicitly describes communication between user interface module 202 and kernel module 206. It also describes “alter[ing]” “the *behavior* of . . . cluster node module 204a . . . *in some embodiments*” so that the cluster node module “can initiate the execution of instructions to activate the remaining cluster node modules 204b–e on the cluster and to complete the sequence to start all kernel modules 206a–e on the cluster” (emphases added). To the extent this alteration of “behavior” somehow limits normal behavior, it only occurs for “some embodiments”—i.e., a subset of “some embodiments” introduced at the beginning of the passage.

The passage verifies that packets *normally* pass from user interface module 202 to kernel module 206a. *See* Ex. 1001, 22:42–67. Therefore, it is only in a subset of the embodiments that packet messages pass from user interface module 202 first, then through cluster node module 204, and finally to kernel module 206a. *See id.* Accordingly, nothing in the passage limits the claims to a direct connection between a user interface and a cluster node module by precluding an intervening kernel, master node, or server. And Patent Owner’s claim construction attempts to allow some forms of indirect communication between cluster nodes, by only attempting to preclude an intervening “*central server*” or “*master node*,” leaving the claim construction open to interpretation without requisite support from the specification. Similarly, Patent Owner explicitly states that its claim

IPR2020-01608
 Patent 8,082,289 B2

construction does not require a direct connection between a user interface and a cluster node module because it allows for intervening “devices” or “components.” *See* Prelim. Resp. 21 n.4 (“To be clear, this construction does **not** require a command to be transmitted directly from the user interface to a cluster node module without passing through other devices, such as routers, switches, or other components.”).

In addition, cluster node module 204a acts as a “master node” or “central server” when it is connected to the user interface module, because messages from other kernels (nodes) must pass through cluster node module 204a on their way to the kernel associated with cluster node module 204a and/or to the user interface node.³ *See, e.g.*, Ex. 1001, Fig. 2, 6:15–19 (“Results of evaluations performed by kernel modules 206a-e are communicated back to the first cluster node module 204a via the cluster node modules 204a-e, which communicates them to the user interface module 208.”), 11:29–32 (“In one embodiment, the cluster node modules 204a-e provide a way for many kernel modules 206a-e such as, for example, Mathematica kernels, running on a computer cluster 100 to communicate with one another.”), 24:15–17 (“The cluster node module creates an illusion that a kernel module is communicating directly with the other kernel modules.”). Cluster node module 204a also acts as a “central server” because it instigates connections to the remaining cluster node modules, according to the column 22 passage discussed and reproduced

³ According to the specification, “[t]he term ‘node’ refers to a processing unit or subunit that is capable of single-threaded execution of code.” Ex. 1001, 4:42–44. The specification also describes “computers, microprocessors, and/or processor cores (‘nodes’).” *Id.* at 1:22–25.

IPR2020-01608
Patent 8,082,289 B2

above. It also controls the other cluster node modules in a “procedure to shut down the system.” *See id.* at 25:34–50.

The specification also indicates that a load balancing embodiment includes a “root processor” that assigns tasks to each of the cluster nodes. *See* Ex. 1001, 21:20–30. On this preliminary record, this further shows that the claims do not require a cluster node module “to exchange messages such that each node can communicate tasks and data with other nodes *without the tasks and data being required to go through a central server or master node*” as asserted by Patent Owner.

The prosecution history also does not support the negative limitations argued by Patent Owner. The Examiner of the application resulting in the ’289 patent issued a single Office Action provisionally rejecting the claims under statutory and non-statutory double patenting. Ex. 1002, 105–06. The Applicant responded by requesting the provisional rejections be withdrawn (*id.* at 118–27), and the Examiner responded by issuing a Notice of Allowance (*id.* at 155–59). Notably, the Applicant expressly stated that no disavowals were made:

Applicants reserve the right to pursue at a later date any previously pending or other broader or narrower claims that capture any subject matter supported by the present disclosure, including subject matter found to be specifically disclaimed herein or by any prior prosecution. *Accordingly, reviewers of this or any parent, child or related prosecution history shall not reasonably infer that Applicants have made any disclaimers or disavowals of any subject matter supported by the present application.*

Id. at 126 (emphasis added).

Based on the foregoing discussion, requisite disclaimer, disavowal, or lexicography does not appear to exist on this preliminary record to import

IPR2020-01608
 Patent 8,082,289 B2

Patent Owner’s proposed negative limitations into the plain language of the challenged claims. *See Omega Eng’g*, 334 F.3d at 1323–26. On this limited record, based on the arguments presented, the parties appear to agree that relatively generic structural implementations of the “kernel module” and “cluster node module” implement the functions recited in the challenged claims without any specific algorithmic structure that the specification may or may not disclose limiting the generic structure.⁴

No other terms require an express construction. Only those terms that are in controversy need be construed, and only to the extent necessary to resolve the controversy. *Nidec*, 868 F.3d at 1017.

D. Overview of the Asserted Prior Art

1. Schreiner1 – Ex. 1008

Schreiner1 is titled “Distributed Maple: parallel computer algebra in networked environments” and bears a copyright date of 2003. Ex. 1008, 3. Schreiner1 is a journal article authored by Dr. Schreiner, Christian

⁴ The term “‘module’ is a well-known nonce word that can operate as a substitute for ‘means’ in the context of § 112, para. 6.” *Williamson v. Citrix Online, LLC*, 792 F.3d 1339, 1349–50 (Fed. Cir. 2015) (en banc) (“[U]se of the word ‘means’ creates a presumption that [35 U.S.C.] § 112, ¶ 6 applies.”; “‘Module’ is a well-known nonce word that can operate as a substitute for ‘means’ in the context of § 112, para. 6.”); *see also* 35 U.S.C. § 112, ¶ 6 (“An element in a claim for a combination may be expressed as a means . . . for performing a specified function without the recital of structure, material, or acts in support thereof, and such claim shall be construed to cover the corresponding structure, material, or acts described in the specification and equivalents thereof.”). Nevertheless, at this stage of the proceeding, neither party argues that the nonce word “module” and any surrounding language falls under § 112, ¶ 6. Accordingly, we do not reach this issue for institution purposes.

IPR2020-01608
 Patent 8,082,289 B2

Mittermaier, and Karoly Bosa. *Id.* Schreiner1 “gives a comprehensive overview on the design and the use of ‘Distributed Maple’, an environment for parallel computer algebra on multiprocessors and heterogeneous computer clusters.” *Id.* Schreiner1 explains that Distributed Maple was developed on the basis of the computer algebra system Maple (*id.*) and that Distributed Maple is built on top of the Maple kernel and does not require any kernel extensions (*id.* at 4). According to Schreiner1, Distributed Maple is “so portable that applications can be executed in many different environments” and “so general that it can be applied to schedule tasks of other computer algebra systems (e.g., Mathematica).” *Id.* Schreiner1 describes Distributed Maple as providing “a programming model which is based on functional/logic/dataflow parallelism” that “allows the creation of a large number of implicitly scheduled tasks with automatic resolution of data dependencies and of globally shared data structures with implicit synchronization.” *Id.* The authors describe using Distributed Maple “to develop the first parallel versions for a number of non-trivial applications from algebraic geometry (parallel curve and surface plotting and parallel neighbourhood analysis).” *Id.* at 5. Schreiner1 discloses that “[t]he user interacts with Distributed Maple via a conventional Maple frontend (text or graphical).” *Id.* at 7. Schreiner1 explains that “[t]he core of Distributed Maple is a scheduler program which is completely independent and even *unaware* of Maple” and Distributed Maple “can in fact embed and schedule tasks from any kind of computation kernels that implement a specific communication protocol.” *Id.* at 8.

Schreiner1 discloses that a Distributed Maple session comprises two components: a scheduler and a Maple interface. Ex. 1008, 8. Figure 1 of

IPR2020-01608
 Patent 8,082,289 B2

Schreiner1, reproduced below, depicts a software architecture for a Distributed Maple session:

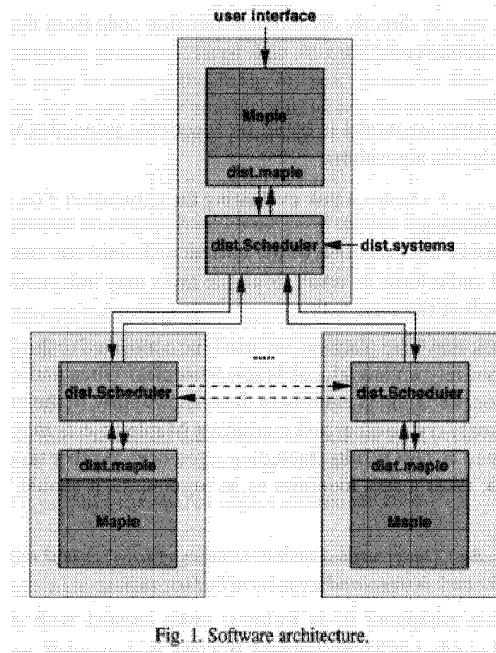


Figure 1 of Schreiner1 illustrates a software architecture for a Distributed Maple session. *Id.* at 9. As shown in Figure 1, a Distributed Maple session “comprises a set of *nodes* each of which holds a pair of processes: a *kernel* and a *scheduler*.” *Id.* at 17. “Initially, a single task runs on the *root* kernel; this task may subsequently create new tasks which are distributed via the schedulers to other kernels and may in turn create new tasks.” *Id.* With reference to Figure 1, Schreiner1 explains that “every scheduler instance accepts tasks from the attached computation kernel and schedules these tasks among all machines connected to the session.” *Id.* at 9. Schreiner1 further explains that “[t]he Maple kernel is a single-threaded process which communicates by a simple communication protocol with the schedule on the same node” and “[a]ll capabilities for parallel and distributed program execution are embedded in this scheduler.” *Id.* at 12.

IPR2020-01608
 Patent 8,082,289 B2

2. *Schreiner2 – Ex. 1009*

Schreiner2 is titled “Distributed Maple – User and Reference Manual (V 1.1.12),” bears a publication date of July 6, 2001, and is authored by Dr. Schreiner. Ex. 1009, 1. Like Schreiner1, Schreiner2 describes the Distributed Maple system. More particularly, Schreiner2 “describes the use of a system for writing distributed Maple applications and sketches its implementation.” *Id.* at 4.

3. *Schreiner3 – Ex. 1010*

Schreiner3 is titled “Task Logging, Rescheduling and Peer Checking in Distributed Maple,” bears a publication date of March 18, 2002, and is authored by Dr. Schreiner and Karoly Bosa. Ex. 1010, 1. Schreiner3 describes extending the Distributed Maple system by adding “fault tolerance mechanisms such that the time spent in a long running computation is not . . . wasted by the eventual occurrence of session failure.” *Id.* Schreiner3 describes a first fault tolerance mechanism as “the logging of task return values and of shared object values such that after a failure the newly started session can (transparently to the application program) reuse already computed result[s].” *Id.* A second fault tolerance mechanism is described as “the migration of tasks such that a session may tolerate the failure of individual nodes without overall failures.” *Id.* A third fault tolerance mechanism is described as “the redirection of the messages such that a session may tolerate also the failure of the connections between nodes without overall failure.” *Id.*

Figure 1 of Schreiner3 illustrates an Execution Model of the Distributed Maple system and is reproduced below:

IPR2020-01608
 Patent 8,082,289 B2

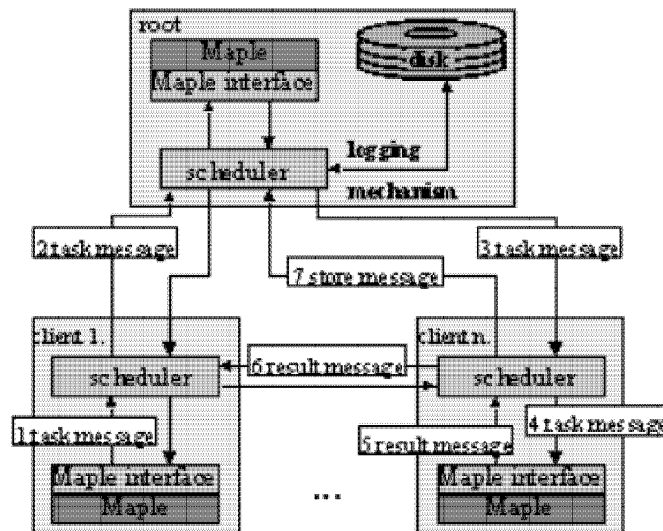


Figure 1: Execution Model

Figure 1 of Schreiner3 illustrates an Execution Model of the Distributed Maple system. Ex. 1010, 5. Figure 1 depicts the passing of messages between nodes in a Distributed Maple system, and a logging mechanism in the root node. *Id.* Schreiner3 explains that “[t]he logging mechanism in Distributed Maple is a fault tolerance mechanism for saving the results of intermediate tasks and the values of shared objects during the computation” and allows the system “to restore the results of computed tasks in a later session, if the current session crashes.” *Id.* at 3.

4. Maple Guide – Ex. 1011

Maple Guide is titled “Maple V Learning Guide, Release 5” published by Waterloo Maple, Inc., and bears a copyright date of 1998. Ex. 1011, 5. Maple Guide explains that “Maple V is a *Symbolic Computation System* or *Computer Algebra System*” and that “[b]oth phrases refer to Maple V’s ability to manipulate information in a symbolic or algebraic manner.” *Id.* at 11.

IPR2020-01608
Patent 8,082,289 B2

5. Distributed Maple Code – Exs. 1012–1018

Distributed Maple Code is a collection of source code for difference components of the Distributed Maple system, the files including: the dist.maple5 file (Ex. 1012); source code for parallel versions of Maple functions in the distsoft directory (Ex. 1013); source code for parallel versions of CASA functions in the distsoft directory (Ex. 1014); an “install” file for Distributed Maple (Ex. 1015); a “readme” file for Distributed Maple (Ex. 1016); an “install” file for the source code in the distsoft directory (Ex. 1017); and a “readme” file for the source code in the distsoft directory (Ex. 1018).

6. Maple Reference – Ex. 1031

Maple Reference is titled “Maple: An Introduction and Reference” and bears a copyright date of 1997. Ex. 1031, 1, 6. Maple Reference is a book that provides an introduction to Maple, and describes the main commands for standard use of Maple and various special commands. *Id.* at 19.

7. Howard – Ex. 1019

Howard is titled “Parallel Processing Systems and Method” and discloses “[m]ethods and systems parallel computation of an algorithm using a plurality of nodes configured as a Howard Cascade.” Ex. 1019, codes (54), (57). “A home node of a Howard Cascade receives a request from a host system to compute an algorithm identified in the request.” *Id.* at code (57). The request is distributed to processing nodes of the Howard Cascade and then participating nodes perform the designated portion of the algorithm in parallel. *Id.* Partial results from each node are agglomerated

IPR2020-01608
Patent 8,082,289 B2

upstream to higher nodes of the structure and then returned to the host system. *Id.*

E. Asserted Obviousness Based on Schreiner1, Schreiner2, Schreiner3, Maple Guide, and Distributed Maple Code

Petitioner argues that claims 1, 4–6, 8, 10, 11, 13, 16–19, 21–23, 27, and 29–32 would have been obvious over the combination of Schreiner1, Schreiner2, Schreiner3, Maple Guide, and Distributed Maple Code.

Pet. 14–77. In support of its showing, Petitioner relies upon the Tufo and Schreiner Declarations. *Id.* (citing Ex. 1005; Ex. 1006). We have reviewed Petitioner’s assertions and supporting evidence. For the reasons discussed below, and based on the record before us, we determine that Petitioner demonstrates a reasonable likelihood of prevailing in showing that these claims would have been obvious over the combination of Schreiner1, Schreiner2, Schreiner3, Maple Guide, and Distributed Maple Code.

1. Claim 1

a. Analysis of Petitioner’s Showing and Patent Owner’s Responses

Petitioner relies on the combined teachings of Schreiner1, Schreiner2, Schreiner3, Maple Guide, and Distributed Maple Code, as supported by the testimony of Dr. Tufo, to allege obviousness of claim 1. Pet. 14–46. As motivation to combine the “Distributed Maple Publications” references, Petitioner contends that they share the same author, Dr. Schreiner, and all relate to the same software project, called “Distributed Maple.” *Id.* at 14–15. Petitioner essentially contends that a person of ordinary skill would have consulted the references to learn details about the system,

IPR2020-01608
Patent 8,082,289 B2

including fault tolerances and capabilities, in order to combine desired features for running the software modules and system. *See id.* at 15.

Regarding the Maple Guide (not authored by Dr. Schreiner), Petitioner asserts that “Schreiner1 teaches that Distributed Maple includes Maple software modules and refers readers to www.maplesoft.com, a website operated by Waterloo Maple, the company that authored and sold the Maple software, for further details.” Pet. 15–16 (citing Ex. 1008, 44;⁵ Ex. 1006 ¶ 49).⁶ Petitioner explains that

Waterloo Maple published the Maple Guide, and a POSITA would have been motivated to read the Maple Guide to learn more about Maple. The teaching in the Distributed Maple Publications that Distributed Maple utilized Maple, including its kernel and libraries, provides a POSITA with a strong, express motivation to combine the features described in the Distributed Maple Publications with the features of Maple, as described in the Maple Guide.

Id. at 16 (citing Ex. 1005 ¶¶ 46–47).

Petitioner also contends that “the [Distributed Maple Publications] references . . . were publicly available on the same webpage – <http://www.risc.uni-linz.ac.at/software/distmaple> – which was cited by Schreiner1 and date-stamped and archived by the Internet Archive, and are submitted as Exhibits 1024 and 1025.” Pet. 15 (citing Ex. 1008, 5–6; Ex. 1009, Abstract, 4; Ex. 1005 ¶ 46; Ex. 1006 ¶¶ 24–25; Ex. 1024;

⁵ Although Petitioner’s citations refer to the journal pagination, we convert the citations to the Exhibit pagination herein.

⁶ Describing Distributed Maple as using “a conventional Maple frontend,” Schreiner1 states that “Maple is a registered of Waterloo Maple Inc.” Ex. 1008, 7. Schreiner1 also cites <http://www.maplesoft.com> under a listing of reference sources, listing “Maple, W., Maple 6, 2001” as one such reference source. *Id.* at 44.

IPR2020-01608
 Patent 8,082,289 B2

Ex. 1025). Schreiner1 states that “[b]oth the Distributed Maple system itself and the library of parallel versions of . . . Maple algorithms are in stable versions freely available under the GNU Library General Public License at <http://www.risc.uni-linz.ac.at/software/distmaple>.” Ex. 1008, 5.

Petitioner maintains that regardless of the above motivation, “Schreiner1 expressly teaches nearly all of the claim limitations by itself, and further motivations to combine for specific features are detailed below in connection with particular claim limitations.” Pet. 16 (citing Ex. 1005 ¶ 48).

i. The Preamble

Claim 1 recites “[a] computer cluster.” Ex. 1001, 28:61. Petitioner asserts that, “[t]o the extent the preamble is limiting, Schreiner1 discloses it.” Pet. 22. Petitioner relies on Schreiner1’s abstract to describe a computer cluster. *Id.* (citing Ex. 1008, Abstract). Petitioner also contends that Schreiner1’s Figure 1 depicts a cluster and Schreiner1 otherwise describes “parallel operations on clusters.” *Id.* (citing Ex. 1008, Fig. 1, 22–42; Ex. 1005 ¶ 61).

Patent Owner does not contest this aspect of the Petition. *See generally* Prelim. Resp.

Schreiner1 explains that it “describe[s] the design and use of Distributed Maple, an environment for executing parallel computer algebra programs on multiprocessors and heterogeneous clusters.” Ex. 1008, Abstract. Schreiner1 further discloses using “a 24 processor heterogeneous computer cluster, an 18-processor Sun HPC 6500 system, and a Linux-based Beowulf cluster with 16 compute nodes linked by two 100 Mbit switched

IPR2020-01608
Patent 8,082,289 B2

Ethernets” and “a 128 processor SGI Origin 3800 distributed shared memory multiprocessor.” *Id.* at 22–23, 25.

Accordingly, for the foregoing reasons and on this preliminary record, to the extent the preamble is limiting, Schreiner1 supports Petitioner’s contentions.

ii. The Processor Recitations

Claim 1 recites “a first processor,” “a second processor,” and “a third processor.” Ex. 1001, 28:62–64. Petitioner argues that “Schreiner1 discusses several implementations of its design, including a 128 processor SGI Origin 3800 distributed shared memory multiprocessor cluster, a 24-processor heterogeneous computer cluster, an 18-processor Sun HPC 6500 system, and a Linux-based Beowulf cluster with 16 compute nodes.” Pet. 22 (emphasis omitted) (citing Ex. 1008, 22–23, 25). Petitioner argues that each of these examples includes three processors. *Id.* (citing Ex. 1005 ¶ 62).

Patent Owner does not contest this aspect of the Petition. *See generally* Prelim. Resp.

As noted above, Schreiner1 discloses using “a 24 processor heterogeneous computer cluster, an 18-processor Sun HPC 6500 system, and a Linux-based Beowulf cluster with 16 compute nodes linked by two 100 Mbit switched Ethernets” and “a 128 processor SGI Origin 3800 distributed shared memory multiprocessor.” Ex. 1008, 22–23, 25. Dr. Tufo states that each of these systems includes three processors. Ex. 1005 ¶ 62.

Accordingly, for the foregoing reasons and on this preliminary record, Schreiner1 supports Petitioner’s contentions.

IPR2020-01608
 Patent 8,082,289 B2

iii. The Computer-Readable Medium Recitation

Claim 1 recites “at least one computer-readable medium in communication at least one of the first processor, the second processor, or the third processor.” Ex. 1001, 28:65–67. Petitioner argues that “Maple is installed in storage and loaded in memory when the program is run by one or more processors.” Pet. 23 (citing Ex. 1011, 15, 102).⁷ Petitioner also contends that “[t]he Maple ‘kernel consists of highly optimized C code.’” *Id.* (quoting Ex. 1011, 98). Petitioner also contends that Distributed Maple is a software program loaded into memory for execution by one or more processors. *Id.* (citing Ex. 1008, 3–5, 7–22; Ex. 1015; Ex. 1016; Ex. 1009, 9; Ex. 1005 ¶ 67). Petitioner also points to “shared memory” as disclosed in Schreiner1 as a computer-readable medium for running Maple and holding the Maple kernels and Distributed Maple code. *Id.* at 24 (citing Ex. 1008, 9, 12, 25; Ex. 1009, 9; Ex. 1005 ¶ 67).

Patent Owner does not contest this aspect of the Petition. *See generally* Prelim. Resp.

As noted above, Schreiner1 discloses using “a 24 processor heterogeneous computer cluster, an 18-processor Sun HPC 6500 system, and a Linux-based Beowulf cluster with 16 compute nodes linked by two 100 Mbit switched Ethernets” and “a 128 processor SGI Origin 3800 distributed shared memory multiprocessor.” Ex. 1008, 22–23, 25. Dr. Tufo opines that “a POSITA would expect the dist.maple and scheduler libraries, as well as copies of the Maple kernel files, to be located in a globally connected storage medium from which they can be loaded to the cluster nodes.”

⁷ Although Petitioner’s citations refer to the book pagination, we convert the citations to the Exhibit pagination herein.

IPR2020-01608
 Patent 8,082,289 B2

Ex. 1005 ¶ 68. Schreiner1 supports this assertion, for example by explaining that “[t]he file dist.maple [is] read by every Maple kernel [and] implements the interface between kernel and scheduler.” Ex. 1008, 8.

Accordingly, for the foregoing reasons and on this preliminary record, Schreiner1 supports Petitioner’s contentions.

iv. The First Kernel Recitation

Claim 1 recites “a first kernel residing in the at least one computer-readable medium, said first kernel configured to translate commands into code for execution on the first processor.” Ex. 1001, 29:1–3. To address these limitations, Petitioner annotates Schreiner1’s Figure 1 as follows (Pet. 26):

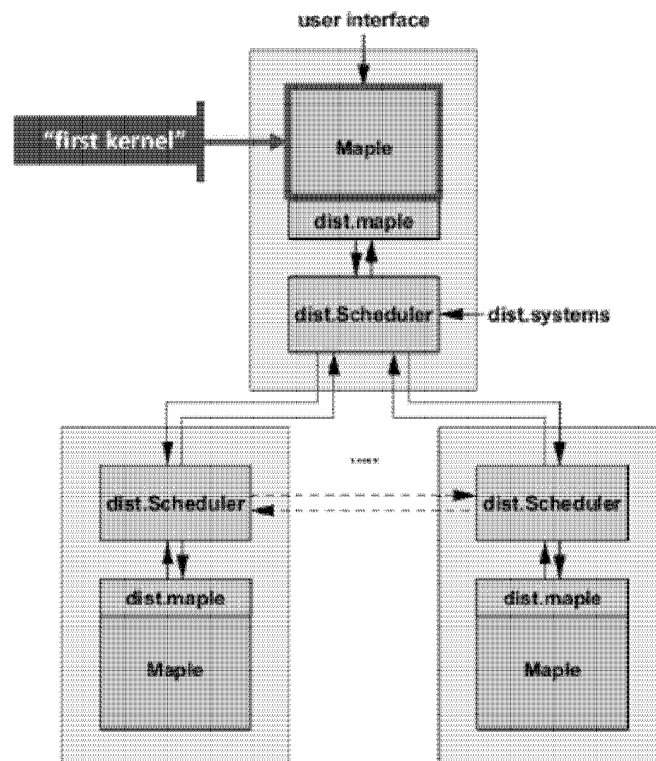


Fig. 1. Software architecture.

IPR2020-01608
Patent 8,082,289 B2

Figure 1 of Schreiner1 illustrates a software architecture for a Distributed Maple session. Ex. 1008, 9. Petitioner has modified this figure above to identify the Maple kernel of the upper node as the recited “first kernel.” Petitioner argues that Distributed Maple connects external computation kernels on various machines and schedules concurrent tasks for execution thereon. Pet. 25 (citing Ex. 1008, 4). Petitioner argues that “Distributed Maple ‘embeds kernels of the computer algebra system Maple as computational engines’ and employs ‘a comparatively high-level programming model.’” *Id.* at 26–27 (quoting Ex. 1008, Abstract).

To support its showing, Petitioner quotes the Maple Guide:

The kernel is the base of Maple’s system. It contains fundamental and primitive commands: *the Maple language interpreter (which converts the commands you type into machine instructions your computer processor can understand)*, algorithms for numerical calculation, and routines to display results and perform other input and output operations.... The Maple kernel implements the most frequently used routines for integer and rational arithmetic and simple polynomial calculations.

Pet. 27 (quoting Ex. 1011, 98; citing Ex. 1005 ¶ 73). Petitioner also explains that “Schreiner2 teaches that high-level commands are translated by the kernel into lower-level code for execution by the processors.” *Id.* (citing Ex. 1009, 7, 15, 30).

Patent Owner does not contest this aspect of the Petition. *See generally* Prelim. Resp.

Schreiner1 explains that Distributed Maple “embeds kernels of the computer algebra system Maple as computational engines into a networked coordination layer implemented in the programming language Java” and “is built on top of . . . the Maple kernel and does not require any kernel

IPR2020-01608
Patent 8,082,289 B2

extensions.” Ex. 1008, Abstract, 4. The Maple Guide explains that the Maple kernel “contains fundamental and primitive commands” including “the Maple language interpreter (which converts the commands you type into machine instructions your computer processor can understand).” Ex. 1011, 98.

According, for the foregoing reasons and on this preliminary record, the asserted references support Petitioner’s contentions. We further determine that, based on this preliminary record, Petitioner has set forth articulated reasoning with rational underpinning explaining why a person of ordinary skill would have combined the teachings of the asserted references. *See* Pet. 14–16 (contending that a person of ordinary skill would have consulted the references to learn details about the Distributed Maple system, including fault tolerances and capabilities, in order to combine desired features for running the software modules and system).

v. The First Cluster Node Module Recitation

Claim 1 recites “a first cluster node module residing in the at least one computer-readable medium, said first cluster node module configured to send commands to the first kernel and receives commands from a user interface.” Ex. 1001, 29:4–7. To address these limitations, Petitioner annotates Schreiner1’s Figure 1 as follows (Pet. 29):

IPR2020-01608
 Patent 8,082,289 B2

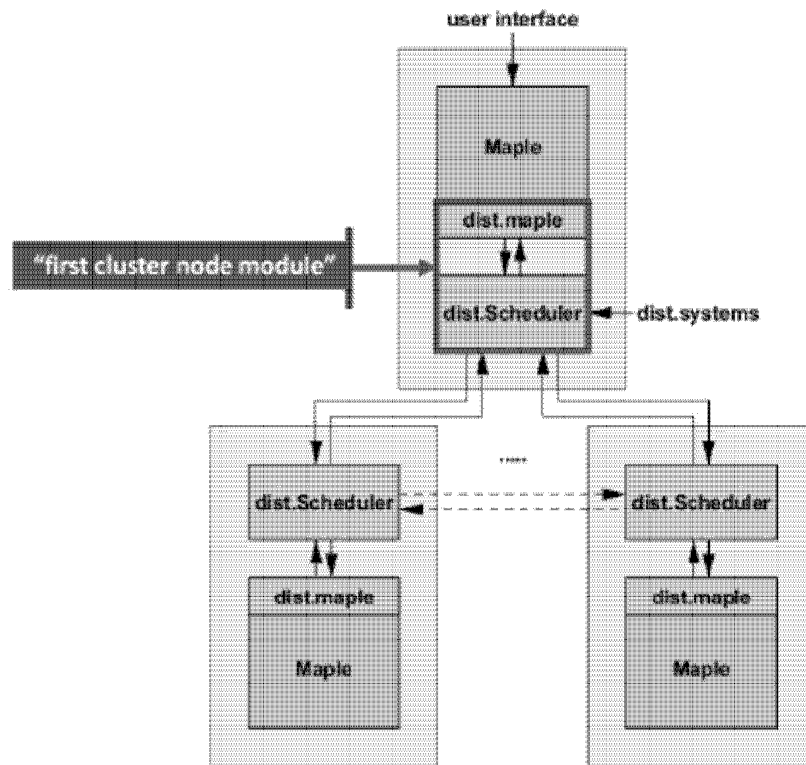


Fig. 1. Software architecture.

Figure 1 of Schreiner1 illustrates a software architecture for a Distributed Maple session. Ex. 1008, 9. Petitioner has modified this figure above to identify the dist.Scheduler Java program and the dist.maple file of the upper node as the recited “first cluster node module.” Petitioner argues that “[t]he dist.Scheduler and dist.maple modules work together to provide communication capabilities: dist.Scheduler ‘coordinates node interaction,’ and dist.maple ‘implements the interface between the kernel and the scheduler.’” Pet. 29–30 (quoting Ex. 1008, 8). Petitioner argues that it would have been obvious for these software modules to reside in the same computer-readable medium because they are accessible by the same processors. *Id.* at 31 (citing Ex. 1005 ¶ 83); *see also id.* at 24 (arguing that Schreiner1 describes an example installation having a “shared memory” that

IPR2020-01608
 Patent 8,082,289 B2

“is a computer-readable medium holding the Maple kernels and Distributed Maple code and in communication with the first, second, and third processors”); Ex. 1005 ¶ 83 (“As software, `dist.maple`, `dist.Scheduler` and the other components of the Distributed Maple cluster node module are accessible by the same processors and reside in the same computer-readable medium, such as shared memory or a shared storage disk, as the corresponding Maple kernel.”). We find, on this preliminary record, that the cited portions of the asserted references and Dr. Tufo’s testimony support Petitioner’s assertions that `dist.Scheduler` and `dist.maple` reside in the same computer-readable medium as the first Maple kernel.

Regarding the recitation that the first cluster node module be configured to send commands to the first kernel, Petitioner argues that “[t]he `dist.Scheduler` component provides ‘[a]ll capabilities for parallel and distributed program execution,’ including sending commands to its ‘attached computation kernel.’” Pet. 31 (second alteration in original) (citing Ex. 1008, 9, 12). Petitioner argues that “[t]he `dist.maple` component ‘implements the interface between kernel and scheduler,’ providing the final link in sending commands to the kernels.” *Id.* (quoting Ex. 1008, 8; citing Ex. 1005 ¶ 84). We find, on this preliminary record, that the cited portions of Schreiner1 support Petitioner’s assertions that `dist.Scheduler` is configured to send commands to the first Maple kernel.

Regarding the recitation that the first cluster node module receives commands from a user interface, Petitioner argues that Schreiner1 explains that “[t]he user interacts with Distributed Maple via a conventional Maple frontend (text or graphical), i.e. she operates within the familiar Maple environment for writing and executing parallel programs.” Pet. 33 (quoting

IPR2020-01608
Patent 8,082,289 B2

Ex. 1008, 7) (citing Ex. 1005 ¶ 89; Ex. 1008, 7–8). We find, on this preliminary record, that the cited portions of Schreiner1 support Petitioner’s assertions that dist.Scheduler and dist.maple receive commands from a user interface.

Patent Owner argues that the Petition fails to explain adequately how the asserted references disclose that the first cluster node module receives commands from the user interface. Prelim. Resp. 31–35. Patent Owner argues that claim 1 requires the first cluster node module to receive commands directly from the user interface without the commands first passing through the first kernel. *Id.* at 31–33. Continuing, Patent Owner argues that the Petition fails to explain adequately how the asserted references disclose cluster node modules. *Id.* at 35–37. Patent Owner reiterates its interpretation of “cluster node module” and asserts that, in the architecture disclosed by Schreiner1, messages are exchanged among the schedulers by the schedulers sending the messages to a root node scheduler that distributes the messages among the machines. *Id.* at 35–36. Patent Owner equates Schreiner1’s root node scheduler to a master node. *Id.* at 36.

Patent Owner’s arguments are premised on its proposed claim construction that effectively requires a direct connection between the user interface and the first cluster node module. As set forth above, the preliminary record does not support Patent Owner’s narrow claim construction. *See* § II.C above.

Accordingly, for the foregoing reasons and on this preliminary record, the asserted references and Dr. Tufo’s testimony support Petitioner’s contentions. We further determine that, based on this preliminary record, Petitioner has set forth articulated reasoning with rational underpinning

IPR2020-01608
 Patent 8,082,289 B2

explaining why a person of ordinary skill would have combined the teachings of the asserted references. *See* Pet. 14–16.

vi. The Second and Third Kernel Recitations

Claim 1 recites “a second kernel residing in the at least one computer-readable medium, said second kernel configured to translate commands into code for execution on the second processor” and “a third kernel residing in the at least one computer-readable medium, said third kernel configured to translate commands into code for execution on the third processor.”

Ex. 1001, 29:8–11, 29:17–20. Referring to Schreiner’s Figure 1, Petitioner maps the Maple kernel of the lower left node to the recited “second kernel” and the Maple kernel of the lower right node to the recited “third kernel.”

Pet. 34–35, 39–40. Petitioner relies on its showing made with respect to the first kernel and argues that the second and third kernels reside on the computer-readable medium and translate commands into code for execution on the second and third processors, respectively. *Id.* at 28, 35, 40.

Patent Owner argues that the Petition fails to explain adequately how the asserted references disclose second and third kernels being configured to translate commands into code. Prelim. Resp. 37–40. Patent Owner argues that the Petition relies solely on the testimony of Petitioner’s declarants, with Dr. Tufo’s testimony being conclusory and Dr. Schreiner’s testimony being based on public use of Distributed Maple rather than a printed publication. *Id.*; *see also* PO Sur-reply 10.

Petitioner argues that “[Dr.] Schreiner’s declaration is based on ‘a series of papers,’ not a public use.” Pet. Reply 9.

We agree with Patent Owner’s assertion that the Petition relies on the testimony of Dr. Tufo and Dr. Schreiner in asserting that, like the first

IPR2020-01608
 Patent 8,082,289 B2

kernel, the second and third kernels are configured to translate commands into code for execution on the respective processors. *See* Pet. 28 (citing Ex. 1005 ¶ 76; Ex. 1006 ¶ 40), 35 (citing Ex. 1005 ¶¶ 93–94), 40 (citing Ex. 1005 ¶¶ 70–72, 102).

Dr. Schreiner discusses how, for “each of the examples discussed in Exhibit 1008, a Distributed Maple cluster was set up” by taking specified actions (Ex. 1006 ¶ 40), which appears to be a discussion of how he used the Distributed Maple cluster. For purposes of institution, we do not consider this and similar portions of the Schreiner Declaration.

Regarding the Tufo Declaration, although Patent Owner only cites to paragraph 76 (*see* Prelim. Resp. 37–38), in which Dr. Tufo states that “[t]hese teachings apply to each kernel in Distributed Maple, with each kernel translating commands into code for execution on its respective processor,” the prior paragraphs of Dr. Tufo’s testimony elucidate the referenced “teachings.” For example, Dr. Tufo relies on Schreiner1, Schreiner2, and the Maple Guide to support his conclusion that “high-level commands,” which “are translated by the kernel into lower-level code for execution by the processors,” are “sent to each of the nodes and then translated by the individual kernels into code causing the *ifactors* library to be loaded into the computer-readable medium.” Ex. 1005 ¶ 74 (citing Ex. 1008, 3; Ex. 1009, 7, 15; Ex. 1011, 98).

On this record, the asserted references support Dr. Tufo’s assertions. For example, the Maple Guide explains that,

When you start Maple, it loads only the *kernel*. The kernel is the base of Maple’s system. It contains fundamental and primitive commands: the Maple language interpreter (which converts the commands you type into machine

IPR2020-01608
 Patent 8,082,289 B2

instructions your computer processor can understand), algorithms for numerical calculation, and routines to display results and perform other input and output operations.

Ex. 1011, 98. Schreiner2 explains that Distributed Maple creates a kernel on each processor in the cluster. Ex. 1009, 7 (discussing “the simple strategy of *data parallelism* where each element of a central input data structure is processed in parallel and the task results are joined to form the desired output structure”), 15 (explaining that the dist[all] command “executes [a command] on *every [Maple] kernel connected to the distributed session*” (emphasis added)). Additionally, Schreiner1 explains that the dist[initialize] command creates Maple kernels on all processors in the cluster. *See* Ex. 1008, 8.

Accordingly, for the foregoing reasons and on this preliminary record, the asserted references and Dr. Tufo’s testimony support Petitioner’s contentions. We further determine that, based on this preliminary record, Petitioner has set forth articulated reasoning with rational underpinning explaining why a person of ordinary skill would have combined the teachings of the asserted references. *See* Pet. 14–16.

vii. The Second and Third Cluster Node Module Recitations

Claim 1 recites “a second cluster node module residing in the at least one computer-readable medium, said second cluster node module configured to send commands to the second kernel and communicates with the first cluster node module” and “a third cluster node module residing in the at least one computer-readable medium, said third cluster node module configured to send commands to the third kernel and configured to communicate with the first cluster node module and the second cluster node

IPR2020-01608
Patent 8,082,289 B2

module.” Ex. 1001, 29:12–16, 29:21–25. Referring to Schreiner1’s Figure 1, Petitioner maps the dist.Scheduler Java program and the dist.maple file of the lower left node to the recited “second cluster node module” and the dist.Scheduler Java program and the dist.maple file of the lower right node to the recited “third cluster node module.” Pet. 35–36, 40–42.

Petitioner relies on its showing made with respect to the first cluster node module and argues that the second and third cluster node modules reside on the computer-readable medium and are configured to send commands to the second and third kernels, respectively. *Id.* Regarding the recitations that the second cluster node module communicates with the first cluster node module and the third cluster node module communicates with the first and second cluster node modules, Petitioner relies on Schreiner1 to disclose that each node within the Distributed Maple cluster can communicate with each of the other nodes. *Id.* at 37 (citing Ex. 1008, 13, 17; Ex. 1005 ¶ 98), 42 (citing Ex. 1008, 13).

Patent Owner does not contest this aspect of the Petition apart from the arguments discussed above with respect to the first cluster node module. *See generally* Prelim. Resp.

Schreiner1 explains that each node connected to a Distributed Maple session includes the dist.Scheduler Java program and the dist.maple file. Ex. 1008, 8–9. Dr. Tufo opines that “a POSITA would expect that the Distributed Maple cluster node module libraries and Maple kernel files would be stored in a global directory on disk or in memory, from which they could be installed on the various cluster nodes.” Ex. 1005 ¶ 96. Schreiner1 explains that each “Maple kernel is a single-threaded process which communicates by a simple communication protocol with the scheduler on

IPR2020-01608
Patent 8,082,289 B2

the same node.” Ex. 1008, 12. Additionally, “all nodes know of each other” and, “[w]hen a node needs to send a message to one of its peers, it can thus establish a direct connection for message transfers.” *Id.* at 13. The dist.Scheduler Java program coordinates this node interaction. *Id.* at 8.

Accordingly, for the foregoing reasons and on this preliminary record, the asserted references and Dr. Tufo’s testimony support Petitioner’s contentions. We further determine that, based on this preliminary record, Petitioner has set forth articulated reasoning with rational underpinning explaining why a person of ordinary skill would have combined the teachings of the asserted references. *See* Pet. 14–16.

viii. The Wherein Recitation

Claim 1 recites “wherein the first cluster node module comprises a data structure in which messages originating from the second and third cluster node modules are stored.” Ex. 1001, 29:26–28. Petitioner argues that the first cluster node module includes a data structure that acts as a buffer to collect messages received from the first and second cluster node modules. Pet. 44–45 (citing Ex. 1005 ¶¶ 108–109; Ex. 1008, 13, Fig. 2; Ex. 1009, 26).

Patent Owner does not contest this aspect of the Petition. *See generally* Prelim. Resp.

Schreiner¹ explains that “all nodes know of each other, i.e. a node knows the address of a machine and the number of a port on which (a thread of) the remote scheduler is listening for connection requests.” Ex. 1008, 13.

The operation of the scheduler is implemented by a number of concurrent threads as shown in Fig. 2. Threads listening on all input channels put the received messages into a central buffer from where a server thread takes them, processes

IPR2020-01608
Patent 8,082,289 B2

them, and creates new messages that are placed in some of the output buffers.

Id. Similarly, Schreiner2 explains that “[a] central server thread sequentially processes messages that were received from any input channel[] and put into a central buffer by a thread listening on that channel.” Ex. 1009, 26.

Accordingly, for the foregoing reasons and on this preliminary record, the asserted references and Dr. Tufo’s testimony support Petitioner’s contentions. We further determine that, based on this preliminary record, Petitioner has set forth articulated reasoning with rational underpinning explaining why a person of ordinary skill would have combined the teachings of the asserted references. *See* Pet. 14–16; *see also id.* at 45–46 (discussing message buffers).

b. Public Accessibility of the Distributed Maple Code

Patent Owner argues, with respect to all of the claim recitations, that Petitioner failed to show that the Distributed Maple Code references were publically accessible prior to the date of the invention. *See* Prelim. Resp. 28–31 (noting that Distributed Maple Code includes Exhibits 1012–1018). However, Petitioner relies on the Distributed Maple Code references, which describe the source code for the Distributed Maple Code referenced in Schreiner1, Schreiner2, Schreiner3, and the Maple Guide, only to support its showing based on the latter references. *See, e.g.*, Pet. 62 (“This understanding is further confirmed in the dist.maple source code files . . .”). As noted above, Petitioner also expressly states that “Schreiner1 expressly teaches nearly all of the claim limitations by itself.” *Id.* at 16 (citing Ex. 1005 ¶ 48).

IPR2020-01608
 Patent 8,082,289 B2

Assuming that the other references do not support institution sufficiently without the source code, Patent Owner contends that Petitioner relies on the uncorroborated testimony of Dr. Schreiner that he posted the Distributed Maple Code (i.e., source code) “in 2003 on the public website of Research Institute for Symbolic Computation (‘RISC’), where the references allegedly could be downloaded through the webpage shown in Exhibit 1024.” Prelim. Resp. 29 (arguing “corroboration is required of a witness’s testimony about his own allegedly invalidating activities” (citing *Finnigan Corp. v. ITC*, 180 F.3d 1354, 1366 (Fed. Cir. 1999))). But Petitioner also cites the website as published in Schreiner1, as Patent Owner acknowledges. *Id.* at 29–30; Ex. 1008, 5, 7. Nevertheless, Patent Owner contends that “[t]he most the evidence submitted by Petitioner shows is that [Dr.] Schreiner himself, or possibly others who helped create the Distributed Maple Code or already knew of its existence, may have been able to locate whatever version was posted at that time.” Prelim. Resp. 31.

This line of argument downplays that Schreiner1, published in the Journal of Symbolic Computation in 2003, would have pointed interested artisans to the website listed therein in order to obtain the “Distributed Maple system itself,” the main subject of Schreiner1 and described as “freely available.” *See* Ex. 1008, 5. Also, the Internet Archive screenshot, Exhibit 1024, describes “Distributed Maple” and lists the same website as published in Schreiner1, and states “Maintained by: Wolfgang Schreiner,” “Last Modification: July 14, 2003.” *See also* Ex. 1006 ¶ 24 (Dr. Schreiner noting that the Internet Archive screenshot states “Last Modification: July 14, 2003” and testifying “[t]hat is consistent with my recollection of the time when I last modified this page” (citing Ex. 1024)); Ex. 1025 (similar

IPR2020-01608
Patent 8,082,289 B2

Internet Archive screenshot evidence). This evidence corroborates Dr. Schreiner's testimony as to the timeframe he uploaded the source code.

Dr. Schreiner also testifies that "[i]t has been my practice to check, from time to time, whether my software was accessible through Google search results, and I did this prior to 2005 for these particular web pages and confirm that my Distributed Maple papers and software were accessible through Google searches." Ex. 1006 ¶ 21. Patent Owner argues that "Petitioner relies entirely upon Schreiner's memory from more than seven years ago to suggest the version of Distributed Maple Code filed in this IPR was publicly accessible back then." Prelim. Resp. 29. However, as indicated above, Schreiner1, coauthored by Dr. Schreiner with two others, and the Internet Archive documents corroborate Dr. Schreiner's testimony about uploading the software on the RISC website.⁸ During trial, Patent Owner will have the opportunity to cross-examine Dr. Schreiner, including regarding Google searches and his memory, assuming for the sake of argument that the ability to search the RISC website using Google is relevant to show public accessibility of the source code.

On this preliminary record, even if Petitioner's showing requires the source code to be publically available to support institution, sufficient evidence exists here to show that the Distributed Maple Code was publically available in 2003 and thereafter up to the date of the invention in 2006. *See* Ex. 1001, code (60) (listing the filing date of a provisional application as October 11, 2006). Moreover, no dispute exists over the fact that some form of the source code existed prior to the date of the invention. This further

⁸ Two others coauthored Schreiner1 with Dr. Schreiner. *See* Ex. 1008, 3 (listing authors "Wolfgang Schreiner, Christian Mittermaier, Karoly Bosa").

IPR2020-01608
 Patent 8,082,289 B2

corroborates that Dr. Schreiner published it on RISC’s public website as Schreiner1 and Exhibit 1024 indicate. Also, to the extent the source code may have changed over the relevant time frame as Patent Owner argues (*see* Prelim. Resp. 29), the parties will have the opportunity to address the materiality of any such changes during trial.

Even if the source code was not publically available at the relevant time, Petitioner’s reliance on it as extrinsic evidence solely to support its showing of how the Distributed Maple system operated at the time of the invention would be proper. *See In re Baxter Travenol Labs.*, 952 F.2d 388, 390 (Fed. Cir. 1991) (explaining that extrinsic evidence may be used to explain what a reference discloses); *Hospira v. Fresenius Kabi USA*, 946 F.3d 1322, 1329 (Fed. Cir. 2020) (“Extrinsic evidence can be used to demonstrate what is ‘necessarily present’ in a prior art embodiment even if the extrinsic evidence is not itself prior art.”). And even if Petitioner must show the public accessibility of such supporting references to rely on such support, on this preliminary record, Petitioner sufficiently shows that Schreiner1, Schreiner2, Schreiner3, and the Maple Guide teach the claim elements without reliance on the supporting source code as disclosed in the Distributed Maple Code.

c. Alleged Objective Evidence of Nonobviousness

Notwithstanding what the teachings of the prior art would have suggested to one skilled in the art, objective evidence of nonobviousness (so called “secondary considerations”) may lead to a conclusion that the challenged claims would not have been obvious. *In re Piasecki*, 745 F.2d 1468, 1471–72 (Fed. Cir. 1984). Objective evidence of nonobviousness “may often be the most probative and cogent evidence in the record” and

IPR2020-01608
 Patent 8,082,289 B2

“may often establish that an invention appearing to have been obvious in light of the prior art was not.” *Transocean Offshore Deepwater Drilling, Inc. v. Maersk Drilling USA, Inc.*, 699 F.3d 1340, 1349 (Fed. Cir. 2012) (citing *Stratoflex, Inc. v. Aeroquip Corp.*, 713 F.2d 1530, 1538 (Fed. Cir. 1983)).

Patent Owner argues that objective evidence regarding its SEM and SET products supports the nonobviousness of the challenged claims. Prelim. Resp. 44–58. Patent Owner puts forth evidence of long-felt and unresolved need, failure by others, praise by others, skepticism of others, and copying. *Id.*

i. Nexus

“In order to accord substantial weight to secondary considerations in an obviousness analysis, ‘the evidence of secondary considerations must have a “nexus” to the claims, i.e., there must be “a legally and factually sufficient connection” between the evidence and the patented invention.’” *Fox Factory, Inc. v. SRAM, LLC*, 944 F.3d 1366, 1373 (Fed. Cir. 2019) (citing *Henny Penny Corp. v. Frymaster LLC*, 938 F.3d 1324, 1332 (Fed. Cir. 2019)). “The patentee bears the burden of showing that a nexus exists” *WMS Gaming Inc. v. Int’l Game Tech.*, 184 F.3d 1339, 1359 (Fed. Cir. 1999). Nexus is a legally and factually sufficient connection between the objective evidence and the claimed invention, such that the objective evidence should be considered in determining nonobviousness. *Demaco Corp. v. F. Von Langsdorff Licensing Ltd.*, 851 F.2d 1387, 1392 (Fed. Cir. 1988). A nexus is presumed when “the patentee shows that the asserted objective evidence is tied to a specific product and that product ‘embodies the claimed features, and is coextensive with them.’” *Fox*

IPR2020-01608
 Patent 8,082,289 B2

Factory, 944 F.3d at 1373 (quoting *Polaris Indus., Inc. v. Arctic Cat, Inc.*, 882 F.3d 1056, 1072 (Fed. Cir. 2018)).

Patent Owner argues that “[t]here is sufficient nexus between the objective evidence related to SEM™ and SET™ and the challenged claims” because the “SEM™ and SET™ products practice at least the challenged independent claims.” Prelim. Resp. 52. Patent Owner argues that “[e]ach of the objective indicia of non-obviousness results from the SEM™ and SET™ architecture embodied by the challenged claims.” *Id.* at 55. Patent Owner argues that “[o]ther parallel-computing architectures failed to meet the long-felt, unmet need precisely because they lacked SEM™ and SET™’s claimed architecture.” *Id.* at 56 (citing Ex. 2006 ¶ 37). Patent Owner argues the “claimed architecture” is also responsible for the other asserted objective indicia of nonobviousness. *Id.* (citing Ex. 2006 ¶¶ 39–42, 46–47).

Thus, Patent Owner’s arguments are premised on its proposed construction for “cluster node module.” *See, e.g.*, Ex. 2006 ¶ 24 (“SEM interposes a communication layer between the front end user interface (view) and the back end Mathematica kernels (model) on each node, and that communication layer manages the peer-to-peer behavior of the nodes.”), 26 (“SET’s architecture interposes a layer in between the front end user interface (view) and a back end kernel (model) that manages the peer-to-peer behavior of the nodes.”), *cited at* Prelim. Resp. 56. However, even if these products fall within the scope of claim 1, for the reasons similar to those explained in § II.C above, the claims do not require the architecture of SEM and SET. In other words, the claims are not architecture-specific. *See MeadWestVaco Corp. v. Rexam Beauty & Closures, Inc.*, 731 F.3d 1258, 1264–65 (Fed. Cir. 2013) (citing *Asyst Techs., Inc. v. Emtrak, Inc.*, 544

IPR2020-01608
Patent 8,082,289 B2

F.3d 1310, 1316 (Fed. Cir. 2008)) (explaining that it was error to consider “secondary considerations of nonobviousness [that] involved only fragrance-specific uses” when “the claims now at issue are not fragrance-specific”). The court in *MeadWestVaco* held that the district court erred because it “credited evidence advanced to show long-felt need and commercial success specific to the perfume industry,” and the claims were not limited to fragrance-specific dispensers. *See id.* (reasoning that “objective evidence of non-obviousness must be commensurate in scope with the claims which the evidence is offered to support” (quoting *Asyst Techs.*, 544 F.3d at 1316)).

For the foregoing reasons and on this preliminary record, Patent Owner fails to meet its burden of establishing a nexus between the objective evidence regarding its SEM and SET products and the claims of the ’289 patent. We, therefore, do not accord substantial weight to such evidence. *Fox Factory*, 944 F.3d at 1373. For the sake of completeness, we nonetheless address below Patent Owner’s allegations relating to objective indicia of nonobviousness.

ii. Long-Felt Need and Failure of Others

“The existence of a long-felt but unsolved need that is met by the claimed invention is . . . objective evidence of non-obviousness.” *Millennium Pharms., Inc. v. Sandoz Inc.*, 862 F.3d 1356, 1369 (Fed. Cir. 2017) (citing *In re Cyclobenzaprine Hydrochloride Extended-Release Capsule Patent Litig.*, 676 F.3d 1063, 1081–83 (Fed. Cir. 2012)). “Long[-]felt need is closely related to the failure of others. Evidence is particularly probative of obviousness when it demonstrates both that a demand existed for the patented invention, and that others tried but failed to satisfy that demand.” *Cyclobenzaprine*, 676 F.3d at 1082.

IPR2020-01608
Patent 8,082,289 B2

Patent Owner argues “there was a long-felt but unmet need for a way to unlock the performance advantages of cluster computing without requiring specialized expertise or excessive time, effort, and cost.” Prelim. Resp. 46. Patent Owner argues that its SEM and SET products met this need. *Id.* at 46–48 (citing Ex. 2001 ¶¶ 83–88; Ex. 2006 ¶¶ 20–30; Ex. 2007 ¶¶ 10, 12; Ex. 2008 ¶¶ 25–28).

Petitioner argues that “[Dr.] Dauger provides no evidence or explanation for why interposing the communications software between the user interface and the kernel, as compared to connecting the communications software in some other way, would make any difference at all to the user.” Pet. Reply 6. Petitioner argues that Dr. Dauger’s testimony is also unpersuasive because it is based on an incorrect claim interpretation requiring the cluster node modules to accept instructions from the user interface without the instructions first passing through any kernel. *Id.* at 6–8.

Patent Owner argues that Petitioner’s arguments regarding claim construction are outside the scope of our Order authorizing Petitioner to file its Reply. PO Sur-reply 6–9. To the contrary, as discussed in considering nexus above (§ II.E.1.c.i), the scope of the claims is relevant to objective indicia of nonobviousness.

Regarding the failure of others, Patent Owner acknowledges that others developed automatic parallelizers and universal compilers that converted serial code to parallel code, but argues that none achieved performance comparable to traditional parallel-computing architectures. Prelim. Resp. 50; *see also* PO Sur-reply 3.

IPR2020-01608
Patent 8,082,289 B2

Petitioner argues that “P[atent] O[wner]’s argument is irrelevant because the claims neither recite ‘automatic’ or ‘universal’ parallelization nor require a specific level of optimization or advantageousness.” Pet. Reply 2 (citing *ABT Sys., LLC v. Emerson Elec. Co.*, 797 F.3d 1350, 1362 (Fed. Cir. 2015)).

Patent Owner’s arguments that a long-felt need existed are based on the testimony of Dr. Dauger, Dr. Bhansali, and Mr. Bancroft. *See* Prelim. Resp. 44–48. However, none of these declarants establishes that others unsuccessfully attempted to solve the problem. Dr. Dauger states that “[n]umerous others had tried to implement automatic parallelizers or universal compilers that would take serial object code as input and output object parallel code,” but “[n]one of these efforts succeeded to produce accurate parallel code that was sufficiently optimized or advantageous enough to catch on.” Ex. 2006 ¶ 37. These conclusory and uncorroborated statements fail to establish that others actually tried to solve the asserted problem. The testimony of the other declarants fares no better. Dr. Bhansali merely states that he was not aware of any other products like the SEM product. Ex. 2007 ¶ 11. Mr. Bancroft states that he “had heard rumours of people trying to develop a universal parallelizer that could be used to automatically parallelize serial code.” Ex. 2008 ¶ 30. None of this testimony persuasively establishes that others actually tried and failed to solve the problem asserted by Patent Owner.

Accordingly, for the foregoing reasons and on this preliminary record, we find Patent Owner’s evidence of long-felt need and failure of others to be weak.

IPR2020-01608
 Patent 8,082,289 B2

iii. Unexpected Results

“If a patent challenger makes a prima facie showing of obviousness, the owner may rebut based on ‘unexpected results’ by demonstrating ‘that the claimed invention exhibits some superior property or advantage that a person of ordinary skill in the relevant art would have found surprising or unexpected.’” *Procter & Gamble Co. v. Teva Pharms. USA, Inc.*, 566 F.3d 989, 994 (Fed. Cir. 2009) (quoting *In re Soni*, 54 F.3d 746, 750 (Fed. Cir. 1995)). “To be particularly probative, evidence of unexpected results must establish that there is a difference between the results obtained and those of the closest prior art, and that the difference would not have been expected by one of ordinary skill in the art at the time of the invention.” *Bristol-Myers Squibb Co. v. Teva Pharms. USA, Inc.*, 752 F.3d 967, 977 (Fed. Cir. 2014); *see also Kao Corp. v. Unilever U.S., Inc.*, 441 F.3d 963, 970 (Fed. Cir. 2006) (“[W]hen unexpected results are used as evidence of nonobviousness, the results must be shown to be unexpected compared with the closest prior art.”) (quoting *Baxter Travenol Labs.*, 952 F.2d at 392).

Patent Owner argues that the performance and ease of use of its SEM and SET products was unexpected. Prelim. Resp. 48–50 (citing Ex. 2006 ¶¶ 38–41). Patent Owner argues that its SEM product outperformed gridMathematica. *Id.* at 48–49. Patent Owner argues that it was able to parallelize Wolfram Research’s Mathematica, Apple’s HD QuickTime Exporter, and Equalis’s Scilab using its SET product in a much shorter time period than expected. *Id.* at 49.

Petitioner argues that Patent Owner’s reliance on the testimony of Dr. Dauger is unpersuasive because Dr. Dauger is a listed inventor of the ’289 patent. Pet. Reply 1. Petitioner also argues that Dr. Dauger’s

IPR2020-01608
 Patent 8,082,289 B2

“testimony is also irrelevant because it compares the claimed invention against gridMathematica, not the ‘closest prior art.’” *Id.* (citing *In re Harris*, 409 F.3d 1339, 1344 (Fed. Cir. 2005); *Trs. of Columbia Univ. v. Illumina, Inc.*, 620 F. App’x 916, 932 (Fed. Cir. 2015)).

Patent Owner replies that “Dr. Dauger’s testimony was submitted under oath and penalty of perjury” and the other evidence cited in its Preliminary Response support its contention that the SEM and SET products exhibited surprising results. PO Sur-reply 1–2. Patent Owner also argues that it “chose the closest prior art by comparing SEM™ with gridMathematica and SET™ with the conventional method of parallelizing applications.” *Id.* at 2 (citing Ex. 2006 ¶ 40).

Patent Owner relies almost exclusively on the testimony of Dr. Dauger in asserting the surprising results of the SEM and SET products. *See* Prelim. Resp. 48–50 (citing Ex. 2006 ¶¶ 38–41). Dr. Dauger testifies the he was surprised that the SEM product performed better than gridMathematica and that Patent Owner was able to parallelize Mathematica “in one man-month.” Ex. 2006 ¶¶ 38–41. Dr. Dauger is an inventor of the ’289 patent, was Patent Owner’s CTO, and is currently a consultant employed by Patent Owner. Ex. 1001, code (75); Ex. 2006 ¶ 1; Ex. 2015, 2; Ex. 2018, 2. On this record, we find Dr. Dauger’s testimony about his personal surprise at the SEM and SET products that he helped create unpersuasive to establish unexpected results of these products. *See In re Cree*, 818 F.3d 694, 702 (Fed. Cir. 2016) (citing *Power-One v. Artesyn Techs., Inc.*, 599 F.3d 1343, 1352 (Fed. Cir. 2010)) (concluding that “self-serving statements from researchers about their own work” do not have the same credibility as statements made by disinterested parties). Notably,

IPR2020-01608
 Patent 8,082,289 B2

Patent Owner provides no evidence to corroborate Dr. Dauger's assertions of unexpected results.

Additionally, Patent Owner provides no comparative testing against any prior art configuration, be it the closest or otherwise. Although Dr. Dauger testifies that some testing was performed (*see* Ex. 2006 ¶ 39), no documentation or data are provided from that testing to substantiate his assertions. This is the type of conclusory evidence that has been found insufficient. *See, e.g., In re Lindner*, 457 F.2d 506, 508 (CCPA 1972) (“This court has said previously that mere lawyers’ arguments unsupported by factual evidence are insufficient to establish unexpected results. . . . Likewise, mere conclusory statements in the specification and affidavits are entitled to little weight when the Patent Office questions the efficacy of those statements.”).

Furthermore, Patent Owner does not persuasively argue that gridMathematica is the closest prior art. As noted by Petitioner, Patent Owner does not compare its products to the asserted references or the Distributed Maple system disclosed therein.

Accordingly, for the foregoing reasons and on this preliminary record, we find Patent Owner's evidence of unexpected results to be weak.

iv. Industry Praise

“Evidence that the industry praised a claimed invention or a product that embodies the patent claims weighs against an assertion that the same claimed invention would have been obvious. Industry participants, especially competitors, are not likely to praise an obvious advance over the known art.” *Apple Inc. v. Samsung Elecs. Co.*, 839 F.3d 1034, 1053 (Fed. Cir. 2016) (en banc).

IPR2020-01608
Patent 8,082,289 B2

Relying on the asserted statements of Dr. Bhansali and Yuko Matsuda, Patent Owner argues that industry praise supports patentability of the '289 patent claims. Prelim. Resp. 50–51. According to Patent Owner, Dr. Bhansali found the SEM product to be efficient for load balancing issues and Mr. Matsuda endorsed the SEM product. *Id.*

Petitioner argues that “[Dr.] Bhansali focuses on ‘load balancing,’” which “has no nexus because the patents do not assert that load balancing was novel or non-obvious.” Pet. Reply 3–4 (citing *Kennametal, Inc. v. Ingersoll Cutting Tool Co.*, 780 F.3d 1376, 1385 (Fed. Cir. 2015)). Petitioner argues that Patent Owner’s proposed construction of “cluster node module” excludes the only method of load balancing disclosed in the '289 patent. *Id.* at 4–5. Petitioner also argues that the references asserted in the Petition teach load balancing. *Id.* at 4.

Patent Owner argues that Petitioner’s arguments regarding claim construction are outside the scope of our Order authorizing Petitioner to file its Reply. PO Sur-reply 1, 4–6. To the contrary, as discussed in considering nexus above (§ II.E.1.c.i), the scope of the claims is relevant to objective indicia of nonobviousness.

In order for evidence of industry praise to be probative of nonobviousness, the evidence must be specifically related to features of the claimed invention. *See Apple*, 839 F.3d at 1053–55 (discussing “substantial evidence of praise in the industry that specifically related to features of the claimed invention”). As argued by Patent Owner, Dr. Bhansali testifies that he found the SEM product to be efficient for load balancing issues, by which he means “*the distribution of different parts of an algorithm or application across different nodes* and the overall process of parallelizing the algorithm

IPR2020-01608
Patent 8,082,289 B2

or application.” Ex. 2007 ¶ 12 (emphasis added). The ’289 patent refers to load balancing in a similar manner. *See* Ex. 1001, 21:8–55. As correctly noted by Petitioner, the challenged claims do not recite load balancing or otherwise require commands to be distributed among the nodes in a particular manner. Thus, on this record, Dr. Bhansali’s testimony appears not to be directed to features of the claimed invention and, therefore, is not probative of nonobviousness.

Regarding the asserted statements made by Mr. Matsuda, Patent Owner cites to the Dauger Declaration rather than any submission endorsed by Mr. Matsuda. Prelim. Resp. 51 (citing Ex. 2006 ¶¶ 44–45). Dr. Dauber cites to a slide deck which he appears to have prepared and the substance of which consists only of two quotations. Ex. 2006 ¶ 44 (citing Ex. 2018, 4); *see also* Ex. 2018, 2 (listing Dean E. Dauger, Ph.D. as the author). On this record, we find the uncorroborated statements of Dr. Dauber, alone, unpersuasive to evidence the asserted statement of Mr. Matsuda.

Dr. Dauger also cites to Exhibit 2023, referring to it as a “white paper” written by Mr. Matsuda. Ex. 2006 ¶ 45 (citing Ex. 2023, 2). Initially, it is not clear what significance a “white paper” carries. Moreover, in the sentence cited by Patent Owner, Mr. Matsuda merely states that the SEM product “stands in an advantageous position” compared with an undefined “Parallel Computing Toolkit” when used with Mathematica. Ex. 2023, 2. This is not the type of competitor praise that courts have found to be indicative of non-obviousness. *See, e.g., Apple*, 839 F.3d at 1053–54 (discussing “numerous internal Samsung documents that both praised Apple’s slide to unlock feature and indicated that Samsung should modify its own phones to incorporate Apple’s slide to unlock feature”).

IPR2020-01608
 Patent 8,082,289 B2

Accordingly, for the foregoing reasons and on this preliminary record, we find Patent Owner’s evidence of the industry praise to be weak.

v. Skepticism

Evidence of industry skepticism weighs in favor of nonobviousness. *See United States v. Adams*, 383 U.S. 39, 52 (1966). “If industry participants or skilled artisans are skeptical about whether or how a problem could be solved or the workability of the claimed solution, it favors nonobviousness.” *WBIP, LLC v. Kohler Co.*, 829 F.3d 1317, 1335 (Fed. Cir. 2016).

Patent Owner argues that experts expressed skepticism that the SEM and SET products would work. Prelim. Resp. 51–52. Regarding the SEM product, Patent Owner relies solely on the testimony of Dr. Bhansali and Mr. Bancroft. *Id.* at 51 (citing Ex. 2007 ¶ 8; Ex. 2008 ¶ 32). Regarding the SET product, Patent Owner relies on the opinion of one unidentified Department of Energy (“DOE”) “reviewer.” *Id.* at 52 (citing Ex. 2006 ¶ 52); *see also* PO Sur-reply 3–4.

Petitioner argues that rather than expressing skepticism that Patent Owner’s products would work, the DOE reviewers in Patent Owner’s exhibits “expressed skepticism over the bold performance claims made by P[atent] O[wner].” Pet. Reply 2–3.

Regarding the SEM product, Patent Owner relies solely on statements of Dr. Bhansali and Mr. Bancroft regarding their personal experience with the SEM product. Prelim. Resp. 51 (citing Ex. 2007 ¶ 8; Ex. 2008 ¶ 32). Dr. Bhansali states that, “[w]hen I first learned about SEM, I was uncertain whether the product would perform as promised.” Ex. 2007 ¶ 8. Dr. Bhansali states that he “graduated from Cal. Tech. with a dual B.S.-M.S.

IPR2020-01608
Patent 8,082,289 B2

in physics, and engineering and applied science” and “received [a] Ph.D. in theoretical physics from Harvard University.” *Id.* ¶¶ 3–4. Neither Patent Owner nor Dr. Bhansali provide any detail about his course of study or industrial experience. Notably, no CV for Dr. Bhansali has been made of record. Accordingly, Patent Owner fails to establish adequately that Dr. Bhansali is a skilled artisan with respect to parallel or distributed computing. Accordingly, we accord Dr. Bhansali’s opinion testimony little weight.

Although Mr. Bancroft states that he provided the SEM product to “many experienced parallel programmers” (Ex. 2008 ¶ 32), no information about or statements made by these asserted experts have been provided. Additionally, neither Patent Owner nor Mr. Bancroft provide his CV, making it difficult to assess his credibility to provide technical testimony. *See* Ex. 2008. We note that Mr. Bancroft’s experience appears to be directed to business development matters rather than technical engineering or computer science research and development. *See id.* ¶¶ 4–14. Moreover, Mr. Bancroft is on Patent Owner’s Business Advisory Board. *Id.* ¶ 33. Thus, it appears that Mr. Bancroft is not a disinterested party and may have economic or other interest in Patent Owner’s success in this proceeding. Accordingly, we accord Mr. Bancroft’s opinion testimony little weight.

Regarding the SET product, we agree that the DOE reviewers appear to indicate that the submissions they reviewed lacked sufficient detail for them to evaluate the performance assertions made in the submissions. Patent Owner relies on the comments of “Reviewer 2” of Exhibit 2019. Prelim. Resp. 52 (citing “Ex. 2006 ¶ 52 (Ex. 2019 at 2)”). This reviewer states, “The proposal . . . provides no quantitative or qualitative evidence of

IPR2020-01608
 Patent 8,082,289 B2

(efficient or not) use of compute[r] resou[r]ces by SET.” Ex. 2019, 2. This reviewer also states, “The applicants have not demonstrated quantitatively that their technology provides real results. . . . SET may prove to be the great success the applicants suggest, but there is no proof that it works on real CAD/CAM/CAE applications.” *Id.* at 3. Continuing, this reviewer states, “The applicants haven’t clearly defined the nature of the plasma code, nor the effort in porting it to SET.” *Id.* Thus, the statements of Reviewer 2 appear to stem from a lack of detail to assess the credibility of the assertions in the submissions.⁹ Other reviewers similarly identify a lack of detail in the submissions. *See id.* at 4 (“[T]he main concepts of this proposal have not been presented in any substantial detail.”; “There is no sound plan to showing that this SET-based approach can be commercially viable.”); Ex. 2021, 1 (“[T]here is no plan to compare the performance of the applications compared to their theoretical performance.”), 2 (“The auto parallelization tools have not provided high performance as they usually have too many generalizations to take advantage of a particular computing architecture. I do not have evidence that the SET tool is any different.”), 4 (“The applicant has provided a general outline of the comparison test approach, but further details in the work plan are needed.”; “While there is an overall projection of technical relevance, the lack of specificity in the proposed test situation presents a high level of uncertainty in achieving the more ambitious goals of this proposal.”); Ex. 2022, 3 (“The performance of SET in Linux and Mac OS operating environments, the type of efficiency increases achieved, and the strength and limitations of the SET approach are

⁹ The submissions made to the DOE are not of record in this case.

IPR2020-01608
Patent 8,082,289 B2

not adequately described.”), 4 (“The applicant has provided a general description of the technical problem and work plan, but specific details of the technical challenges to be encountered with the SET technology should be described in greater detail.”). Thus, to the extent one reviewer expressed skepticism that the SET product would perform as claimed, this evidence is undercut by the overwhelming expression of a lack of detail provided in the submitted proposals that were reviewed.

Accordingly, for the foregoing reasons and on this preliminary record, we find Patent Owner’s evidence of skepticism of others to be weak.

vi. Copying

“Copying may indeed be another form of flattering praise for inventive features.” *Crocs, Inc. v. ITC*, 598 F.3d 1294, 1311 (Fed. Cir. 2010). Copying “requires evidence of efforts to replicate a specific product.” *Wyers v. Master Lock Co.*, 616 F.3d 1231, 1246 (Fed. Cir. 2010). “This may be demonstrated either through internal documents; direct evidence such as disassembling a patented prototype, photographing its features, and using the photograph as a blueprint to build a virtually identical replica; or access to, and substantial similarity to, the patented product (as opposed to the patent).” *Iron Grip Barbell Co. v. USA Sports, Inc.*, 392 F.3d 1317, 1325 (Fed. Cir. 2004) (internal citations omitted). “We note, however, that a showing of copying is only equivocal evidence of nonobviousness in the absence of more compelling objective indicia of other secondary considerations.” *Ecolochem, Inc. v. S. Cal. Edison Co.*, 227 F.3d 1361, 1380 (Fed. Cir. 2000); *see also In re GPAC*, 57 F.3d at 1580 (“[M]ore than the mere fact of copying by an accused infringer is needed to make that action

IPR2020-01608
Patent 8,082,289 B2

significant to a determination of the obviousness issue.” (quoting *Cable Elec. Prods. v. Genmark, Inc.*, 770 F.2d 1015, 1028 (Fed. Cir. 1985))).

Patent Owner asserts that it provided information to Petitioner regarding its SET product during a November 2012 meeting and in a subsequent “email attaching a specially tailored data sheet.” Prelim. Resp. 56 (citing Ex. 2006 ¶ 56). Patent Owner argues that “Petitioner then copied the claimed invention by incorporating the claimed architecture into Petitioner’s GPGPUs in the manner described by the datasheet” and “named its GPU interconnect architecture, which uses the claimed structure, NVLink™.”¹⁰ *Id.* at 57 (citing Ex. 2006 ¶ 59); *see also* PO Sur-reply 9–10.

Petitioner traverses Patent Owner’s assertions of copying, calling it “a gross misrepresentation to the PTAB.” Pet. Reply 8–9. Petitioner asserts that the inventors of the ’289 patent sent an unsolicited email to one of its employees attaching a public SET datasheet that “did not have any suggestion of ‘provid[ing] a communications infrastructure for direct all-to-all communications between each GPU.’” *Id.* at 8 (alteration in original) (citing Prelim. Resp. 56).

Patent Owner provides no evidence to support its assertion. Patent Owner does not provide any description of the NVLink product or compare this product to the claims of the ’289 patent. On this record, Patent Owner’s conclusory assertions are inadequate to establish any copying of the claimed invention by Petitioner.

¹⁰ Dr. Dauger refers to NVIDIA’s “general-purpose GPU (‘GPGPU’) supercomputing” and describes “NVIDIA’S Tesla GPGPUs as hardware black boxes on which the back end executes.” Ex. 2006 ¶¶ 56–57 (citing Ex. 2020, 4, Fig 3).

IPR2020-01608
 Patent 8,082,289 B2

Accordingly, for the foregoing reasons and on this preliminary record, we find Patent Owner's evidence of copying to be weak.

d. Conclusion

As set forth above, we have considered the scope and content of the prior art (§ II.D above), the differences between the claimed subject matter and the prior art (§ **Error! Reference source not found.**), the level of skill in the art (§ II.B above), and the objective evidence of nonobviousness (§ II.E.1.c above). Based on our findings, at this stage of the proceeding, we determine that Petitioner has established a reasonable likelihood of prevailing on its assertion that claim 1 would have been obvious over the combination of Schreiner1, Schreiner2, Schreiner3, Maple Guide, and Distributed Maple Code.

2. Claims 4–6, 8, 10, 11, 13, and 16

Each of claims 4–6, 8, 10, 11, 13, and 16 depends, directly or indirectly, from claim 1. The Petition maps the additional limitations of these challenged dependent claims to Schreiner1, Schreiner2, and Schreiner3. Pet. 46–55. Patent Owner does not challenge separately the arguments and evidence presented for the dependent claims. *See generally* Prelim. Resp. Based on our review of the current record before us, we determine that the information presented in the Petition establishes that there is a reasonable likelihood that Petitioner would prevail in challenging claims 4–6, 8, 10, 11, 13, and 16 as being unpatentable over the combination of Schreiner1, Schreiner2, Schreiner3, Maple Guide, and Distributed Maple Code.

IPR2020-01608
Patent 8,082,289 B2

3. Claim 17

Independent claim 17 recites a computer cluster that is similar to that recited in claim 1. Ex. 1001, 30:23–39. Petitioner relies on the asserted references in a similar manner as advanced for claim 1 and relies on the same rationale for their combination. Pet. 14–16, 56–66.

a. The Preamble

Claim 17 recites “[a] computer cluster.” Ex. 1001, 30:23. Petitioner relies on its showing regarding the preamble of claim 1. Pet. 56. Patent Owner does not contest this aspect of the Petition. *See generally* Prelim. Resp. For the reasons set forth above and on this preliminary record, to the extent the preamble is limiting, the asserted references support Petitioner’s contentions.

b. The Nodes Recitation

Claim 17 recites “plurality of nodes, wherein each node is configured to access a computer-readable medium comprising program code for a user interface and program code for a single-node kernel module configured to interpret user instructions.” Ex. 1001, 30:24–28. Petitioner relies on its showing regarding the Computer-Readable Medium and the First Kernel Recitations of claim 1. Pet. 56–57. Petitioner additionally argues that Schreiner¹ “teaches that Maple was installed on each node of the Distributed Maple cluster.” *Id.* at 57 (citing Ex. 1008, 8–9).

Patent Owner challenges Petitioner’s showings with substantially the same arguments advanced for claim 1. Prelim. Resp. 42. These arguments fail to persuade us to deny institution for the reasons set forth above.

IPR2020-01608
Patent 8,082,289 B2

For the reasons set forth above and on this preliminary record, the asserted references support Petitioner's contentions.

c. The Cluster Node Modules Recitation

Claim 17 recites,

a plurality of cluster node modules, wherein each cluster node module is configured to communicate with a single-node kernel and with one or more other cluster node modules, to accept instructions from the user interface, and to interpret at least some of the user instructions such that the plurality of cluster node modules communicate with one another in order to act as a cluster.

Ex. 1001, 30:29–36. Petitioner relies on its showing regarding the First, Second, and Third Cluster Node Module Recitations of claim 1. Pet. 57–59. In addition, Petitioner annotates Schreiner1's Figure 1 as follows (*id.* at 60):

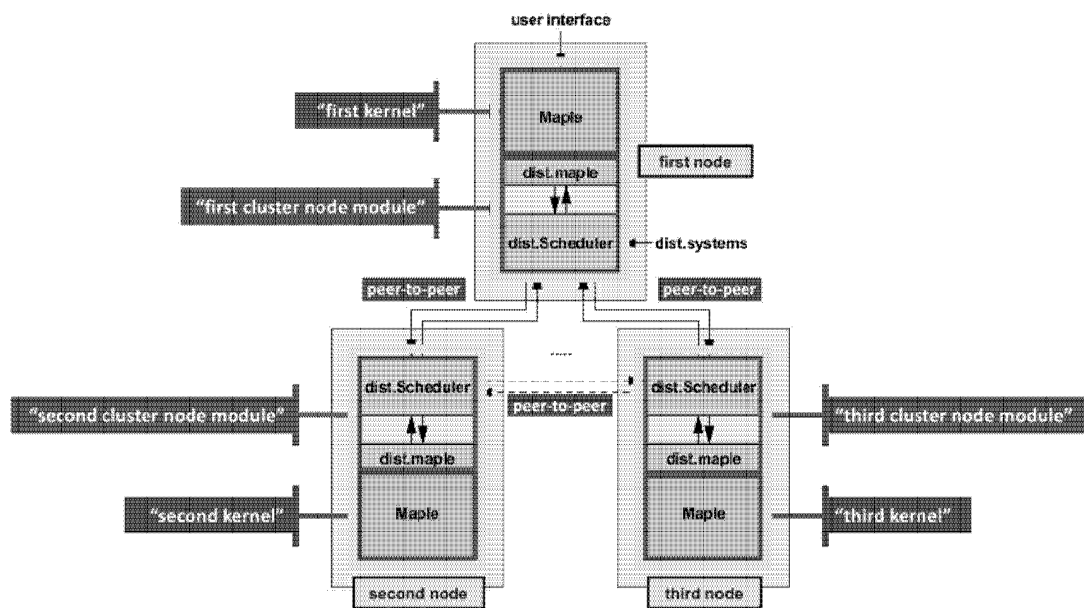


Fig. 1. Software architecture.

Figure 1 of Schreiner1 illustrates a software architecture for a Distributed Maple session. Ex. 1008, 9. Petitioner has modified this figure above to identify the recited kernels and cluster node modules as set forth with

IPR2020-01608
 Patent 8,082,289 B2

claim 1 and to illustrate communication among the nodes with arrows. Pet. 60; *see also id.* at 58 (stating, with respect to a similarly annotated version of Schreiner1’s Figure 1, “cluster node module to kernel communications shown by bidirectional arrows between Maple kernels and their respective scheduler processes, and cluster module-to-module communications shown by arrows between nodes”). Petitioner argues that Schreiner1 discloses a root cluster node module (the labeled “first node”) having a root kernel (the labeled “first kernel”) that receives commands from the user interface and passes them to the first cluster node module (the dist.Maple and dist.Scheduler of the labeled “first node”). *Id.* at 60 (citing Ex. 1008, 3–4, 7–8; Ex. 1005 ¶ 142). Petitioner argues that the commands are then communicated by the root scheduler (the dist.Scheduler of the labeled “first node”) to the other nodes. *Id.* at 61 (citing Ex. 1008, 8–9, 13).

Regarding the recitation that the cluster node modules are configured to interpret at least some of the user instructions, Petitioner argues that “the cluster node modules interpret the dist[all] command such that the nodes communicate to evaluate a Maple statement in parallel.” Pet. 61–62 (citing Ex. 1005 ¶ 144). Petitioner also argues that the cluster node modules interpret other commands, such as dist[start] and ssiPilot, to act as a cluster and evaluate expressions in parallel. *Id.* at 63 (citing Ex. 1008, 4, 5, 8, 10–12; Ex. 1005 ¶¶ 146–148).

Patent Owner challenges Petitioner’s showings with substantially the same arguments advanced for claim 1. Prelim. Resp. 40–41. These arguments fail to undermine Petitioner’s sufficient showing for purposes of institution for the reasons set forth above.

IPR2020-01608
Patent 8,082,289 B2

As noted in § II.E.1.a.v above, on this preliminary record Petitioner has persuasively explained how Schreiner1 discloses cluster node modules that accept instructions from a user interface. Schreiner1 further explains that “dist[all] (*command*) lets the Maple statement *command* be executed on every Maple kernel connected to the distributed session” and dist[start] creates tasks that are scheduled on a number of Maple kernels.

Ex. 1008, 9–10.

According, for the foregoing reasons and on this preliminary record, Schreiner1 supports Petitioner’s contentions.

d. The Communication Network Recitation

Claim 17 recites “a communications network to connect the nodes.” Ex. 1001, 30:37. Petitioner notes that Schreiner1 is titled “Distributed Maple: parallel computer algebra in **networked environments**” and argues that Schreiner1 “describes numerous examples of communications networks connecting the nodes.” Pet. 46–47 (citing Ex. 1008, 3, 16–17, 23, 25, 35); *see also id.* at 64 (citing *id.* at 46–47). Patent Owner does not contest this aspect of the Petition. *See generally* Prelim. Resp. We find, on this preliminary record, that the cited portions of Schreiner1 support Petitioner’s assertions that the nodes are connected by a communication network.

e. The Wherein Recitation

Claim 17 recites “wherein one of the plurality of cluster node modules returns a result to the user interface.” Ex. 1001, 30:38–39. Petitioner argues that “Schreiner1 teaches that the cluster node modules return results to the user interface via the root node” using the dist[start] and dist[wait] commands. Pet. 64–65 (citing Ex. 1008, 8, 10, 14). Petitioner reproduces

IPR2020-01608
 Patent 8,082,289 B2

examples of textual and graphical results sent to the user interface. *Id.* at 65 (citing Ex. 1008, 8, 28).

Patent Owner does not contest this aspect of the Petition. *See generally* Prelim. Resp.

Schreiner1 supports Petitioner's contentions. For example, Schreiner1 explains that, "After the distributed session has been successfully established, two calls of dist[start] create two tasks evaluating the Maple expressions $\text{int}(x^n, x)$ and $\text{int}(x^n, n)$, respectively. The two dist[wait] calls block the current execution until the corresponding tasks have terminated and then return their results." Ex. 1008, 8. As noted by Petitioner, the results can be returned to the user graphically as a plot. *See id.* at 28.

f. Conclusion

As set forth above, we have considered the scope and content of the prior art (§ II.D above), the differences between the claimed subject matter and the prior art (§§ II.E.3.a–e above), the level of skill in the art (§ II.B above), and the objective evidence of nonobviousness (§ II.E.1.c above). Based on our findings, at this stage of the proceeding, we determine that Petitioner has established a reasonable likelihood of prevailing on its assertion that claim 17 would have been obvious over the combination of Schreiner1, Schreiner2, Schreiner3, Maple Guide, and Distributed Maple Code.

4. Claims 18, 19, 21–23, and 27

Each of claims 18, 19, 21–23, and 27 depends, directly or indirectly, from claim 17. The Petition maps the additional limitations of these

IPR2020-01608
Patent 8,082,289 B2

challenged dependent claims to Schreiner1 and Schreiner3. Pet. 66–72. Patent Owner does not challenge separately the arguments and evidence presented for the dependent claims. *See generally* Prelim. Resp. Based on our review of the current record before us, we determine that the information presented in the Petition establishes that there is a reasonable likelihood that Petitioner would prevail in challenging claims 18, 19, 21–23, and 27 as being unpatentable over the combination of Schreiner1, Schreiner2, Schreiner3, Maple Guide, and Distributed Maple Code.

5. Claim 29

Independent claim 29 recites a method of evaluating a command on a computer cluster having recitations that are similar to those of claims 1 and 17. Ex. 1001, 31:17–31. Petitioner relies on the asserted references in a similar manner as advanced for claims 1 and 17 and relies on the same rationale for their combination. Pet. 14–16, 72–75.

a. The Preamble

Claim 29 recites “[a] method of evaluating a command on a computer cluster.” Ex. 1001, 31:17–18. Petitioner relies on its showing regarding the preamble, First Kernel, and First Cluster Node Recitations of claim 1. Pet. 72. Patent Owner does not contest this aspect of the Petition. *See generally* Prelim. Resp. For the reasons set forth above and on this preliminary record, to the extent the preamble is limiting, the asserted references supports Petitioner’s contentions.

b. The Command Communicating Recitation

Claim 29 recites “communicating a command from at least one of a user interface or a script to one or more cluster node modules within the

IPR2020-01608
Patent 8,082,289 B2

computer cluster.” Ex. 1001, 31:19–21. Petitioner relies on its showing regarding the First, Second, and Third Cluster Node Module Recitations of claim 1. Pet. 72.

Patent Owner challenges Petitioner’s showings with substantially the same arguments advanced for claims 1 and 17. Prelim. Resp. 42–44. These arguments fail to persuade us to deny institution for the reasons set forth above.

For the reasons set forth above and on this preliminary record, the asserted references support Petitioner’s contentions.

c. The Message Communicating Recitation

Claim 29 recites “for each of the one or more cluster node modules, communicating a message based on the command to a respective kernel module associated with the cluster node module.” Ex. 1001, 31:22–25. Petitioner relies on its showing regarding the First, Second, and Third Cluster Node Module Recitations of claim 1. Pet. 73.

Patent Owner challenges Petitioner’s showings with substantially the same arguments advanced for claims 1 and 17. Prelim. Resp. 42–44. These arguments fail to persuade us to deny institution for the reasons set forth above.

For the reasons set forth above and on this preliminary record, the asserted references supports Petitioner’s contentions.

d. The Receiving Recitation

Claim 29 recites “for each of the one or more cluster node modules, receiving a result from the respective kernel module associated with the cluster node module.” Ex. 1001, 31:26–28. Petitioner relies on its showing

IPR2020-01608
Patent 8,082,289 B2

regarding the Wherein Recitation of claim 1. Pet. 73. Patent Owner does not contest this aspect of the Petition. *See generally* Prelim. Resp. For the reasons set forth above and on this preliminary record, the asserted references supports Petitioner’s contentions.

e. The Responding Recitation

Claim 29 recites “for at least one of the one or more cluster node modules, responding to messages from other cluster node modules.” Ex. 1001, 31:29–31. Petitioner argues that Schreiner1 discloses this recitation. Pet. 73–75. For example, Petitioner argues that Schreiner1 discloses

an example where node n sends a task message to node n’ for execution, node n’’ asks node n for the result of that task, node n’ responds with the result of the task to node n (which is one instance of the claimed “responding”), and node n then sends a reply to node n’’ containing the result (another instance of the claimed “responding”).

Id. at 73–74 (citing Ex. 1008, 18, Fig. 5; Ex. 1005 ¶ 73). Patent Owner does not contest this aspect of the Petition. *See generally* Prelim. Resp. We find, on this preliminary record, that the cited portions of Schreiner1 support Petitioner’s assertions that the nodes respond to messages from other cluster node modules.

f. Conclusion

As set forth above, we have considered the scope and content of the prior art (§ II.D above), the differences between the claimed subject matter and the prior art (§§ II.E.5.a–e above), the level of skill in the art (§ II.B above), and the objective evidence of nonobviousness (§ II.E.1.c above). Based on our findings, at this stage of the proceeding, we determine that

IPR2020-01608
 Patent 8,082,289 B2

Petitioner has established a reasonable likelihood of prevailing on its assertion that claim 29 would have been obvious over the combination of Schreiner1, Schreiner2, Schreiner3, Maple Guide, and Distributed Maple Code.

6. Claims 30–32

Each of claims 30–32 depends directly from claim 29. The Petition maps the additional limitations of these challenged dependent claims to Schreiner1, Schreiner2, and Schreiner3. Pet. 75–77. Patent Owner does not challenge separately the arguments and evidence presented for the dependent claims. *See generally* Prelim. Resp. Based on our review of the current record before us, we determine that the information presented in the Petition establishes that there is a reasonable likelihood that Petitioner would prevail in challenging claims 30–32 as being unpatentable over the combination of Schreiner1, Schreiner2, Schreiner3, Maple Guide, and Distributed Maple Code.

F. Asserted Obviousness Based on Schreiner1, Schreiner2, Schreiner3, Maple Guide, Distributed Maple Code, Maple Reference, and Howard

Petitioner argues that claim 14 would have been obvious the combination of Schreiner1, Schreiner2, Schreiner3, Maple Guide, Distributed Maple Code, Maple Reference, and Howard. Pet. 77–82. In support of its showing, Petitioner relies upon the Tufo Declaration. *Id.* (citing Ex. 1005). We have reviewed Petitioner’s assertions and supporting evidence. For the reasons discussed below, and based on the record before us, we determine that Petitioner demonstrates a reasonable likelihood of prevailing in showing that this claim would have been obvious over the

IPR2020-01608
 Patent 8,082,289 B2

combination of Schreiner1, Schreiner2, Schreiner3, Maple Guide, Distributed Maple Code, Maple Reference, and Howard.

Claim 14 depends indirectly from claim 1 through claim 13 and further recites “wherein the advanced functions module comprises a call that calculates a Fourier transform across the computer cluster in parallel.” Ex. 1001, 30:13–15. Petitioner argues that, because Schreiner1 discloses “a toolkit of advanced parallelized functions,” it would have been obvious to include “calculating a Fourier transform across the computer cluster in parallel,” as taught by Howard, in Distributed Maple. Pet. 78 (citing Ex. 1019 ¶¶ 141–152, 437–447, Figs. 66–69; Ex. 1005 ¶ 198). Petitioner argues that including Fourier transforms in the library of Distributed Maple functions would have been obvious because, during prosecution of an descendant patent of the ’289 patent, the Examiner rejected a claim reciting the parallelized performance of Fourier transforms by finding that Howard taught the Fourier transform limitations. *Id.* at 79 (citing Ex. 1021, 384–85). Thus, according to Petitioner, “the USPTO already has issued a finding combining the same teachings of Howard used in this Petition—and that finding was not refuted by the applicant during prosecution.” *Id.* (citing Ex. 1021, 428–29; Ex. 1005 ¶ 205). Petitioner additionally argues that it would have been obvious to perform Fourier transforms using Distributed Maple because Fourier transforms are a standard tool in many fields with well-known practical importance. *Id.* at 80–82.

Patent Owner does not challenge separately the arguments and evidence presented for the dependent claims. *See generally* Prelim. Resp.

Howard supports Petitioner’s assertions. For example, Howard discloses that a two-dimensional fast Fourier transform (“FFT”) is computed

IPR2020-01608
 Patent 8,082,289 B2

on a data set consisting of an array having m rows and n columns (such as a bitmap image) by performing a one-dimensional FFT on each dimension. Ex. 1019 ¶ 142. Then, the m rows and n columns are distributed over the parallel processing nodes (*id.* ¶¶ 143–145), which compute the one-dimensional FFTs (*id.* ¶ 148). The results are accumulated in a home node to produce the final result. *Id.* We further determine that, based on this preliminary record, Petitioner has set forth articulated reasoning with rational underpinning explaining why a person of ordinary skill would have included Fourier transforms in the library of Distributed Maple functions.

According, based on our findings and on this preliminary record, we determine that Petitioner has established a reasonable likelihood of prevailing on its assertion that claim 14 would have been obvious over the combination of Schreiner1, Schreiner2, Schreiner3, Maple Guide, Distributed Maple Code, Maple Reference, and Howard.

G. Patent Owner’s Additional Arguments

Patent Owner presents a number of additional arguments asserting that we should deny institution. Prelim. Resp. 58–66. First, Patent Owner argues that “[t]he Petition violates the requirement to construe the claims because it does not construe *any* claim terms, even though construction is necessary to resolve the Petition.” *Id.* at 58. Patent Owner argues that we should deny institution because Petitioner failed to address, in its claim construction section, “cluster node module” and how the cluster node modules receive commands from the user interface. *Id.* at 59–61.

We are not persuaded that we should exercise our discretion to deny institution based on Petitioner’s alleged failure to set forth adequate claim

IPR2020-01608
Patent 8,082,289 B2

constructions. We understand Petitioner’s statement that “[t]he Challenged Claims are interpreted using the same standard used in federal district court” (Pet. 13 (citing 37 C.F.R. § 42.100(b))) to assert that the claims terms should be construed according to their ordinary and customary meaning. Thus, for purposes of this Decision, Petitioner has complied with our rule that the Petition must identify how the challenged claims are to be construed. *See* 37 C.F.R. § 42.104(b)(3).

Next, Patent Owner reiterates its argument that the Petition relies on assertions of public use made by Dr. Schreiner to fill in gaps in the asserted references. Prelim. Resp. 61–65. Patent Owner argues that we must deny institution if the Petition relies on public use in its challenges to the ’289 patent. *Id.* at 62. Patent Owner also argues that even if we were to determine that the Petition does not rely improperly on asserted public use, we should exercise our discretion to deny institution “to avoid the injustice of allowing Petitioner to have two bites at the apple” because it is uncertain whether a district court “would apply estoppel to bar Petitioner from re-litigating essentially the same invalidity challenge in the form of a public use challenge.” *Id.* at 63.

We are not persuaded that we should exercise our discretion to deny institution. As explained above, to the extent that Petitioner relies on Dr. Schreiner’s assertions of public use, we do not consider such testimony for purposes of this Decision. As also explained above, we conclude that Petitioner maps the challenged claims to the asserted references adequately to satisfy its burden to establish a reasonable likelihood that Petitioner would prevail with respect to at least one of the claims challenged in the Petition.

IPR2020-01608
Patent 8,082,289 B2

Nor are Patent Owner's assertions that instituting *inter partes* review would improperly allow Petitioner "two bites at the apple." *See* Prelim. Resp. 63. First, as explained above, to the extent Petitioner makes arguments based on asserted public use, we do not consider such arguments. Moreover, Patent Owner's arguments amount to mere speculation about future events in a separate proceeding before a different tribunal. Such speculation does not persuade us that we should deny institution.

Finally, Patent Owner argues that "the Petition used tricks to undercount words," such as omitting spaces in abbreviations in citations to the papers and exhibits and copying text as images. Prelim. Resp. 66. According to Patent Owner, "[t]hese instances add up to about 1,000 excess words." *Id.* Patent Owner argues "[t]he Board has rejected briefs that used similar tricks." *Id.* (citing *Starbucks Corp. v. Ameranth, Inc.*, CBM2015-00091, Paper 16 at 2–3 (PTAB Jan. 29, 2016)).

Patent Owner does not seek a specific remedy. Rather, Patent Owner generally seeks a denial of institution at the end of its Preliminary Response "[f]or [all of] the foregoing reasons." *See* Prelim. Resp. 66 (citing *Starbucks*, Paper 16 (Board order directing patent owner to re-file its brief, which Patent Owner characterizes as a "rejected brief")). In any event, we are not persuaded that Petitioner's omission of spaces from its citation abbreviations amounts to "tricks" or gamesmanship warranting the re-filing of the Petition at this late stage or denial of institution. We agree with Patent Owner that the Petition incorporates text as images in three instances. *See* Prelim. Resp. 66 (citing Pet. 21, 54, 65). Although Patent Owner does not identify the number of "excess words" resulting from these images, it appears that there are approximately 280 words contained in these images.

IPR2020-01608
Patent 8,082,289 B2

We are not persuaded that the improper inclusion of this number of words as images or Petitioner's omission of these words from the word count warrants the re-filing of the Petition at this late stage or the draconian remedy of denying institution, especially where Patent Owner does not seek a specific remedy.

III. CONCLUSION

For the foregoing reasons, we determine that the information presented establishes a reasonable likelihood that Petitioner would prevail in showing that at least one of claims 1, 4–6, 8, 10, 11, 13, 14, 16–19, 21–23, 27, and 29–32 of the '289 patent is unpatentable.

Our determination in this Decision is not a final determination on either the patentability of any challenged claim or the construction of any claim term and, thus, leaves undecided any remaining fact issues necessary to determine whether sufficient evidence supports Petitioner's contentions by a preponderance of the evidence in the final written decision. *See TriVascular, Inc. v. Samuels*, 812 F.3d 1056, 1068 (Fed. Cir. 2016) (noting that “there is a significant difference between a petitioner's burden to establish a ‘reasonable likelihood of success’ at institution, and actually proving invalidity by a preponderance of the evidence at trial”).

IV. ORDER

Accordingly, it is:

ORDERED that pursuant to 35 U.S.C. § 314(a), an *inter partes* review of claims 1, 4–6, 8, 10, 11, 13, 14, 16–19, 21–23, 27, and 29–32 of

IPR2020-01608
Patent 8,082,289 B2

the '289 patent is instituted with respect to all grounds set forth in the Petition; and

FURTHER ORDERED that, pursuant to 35 U.S.C. § 314(c) and 37 C.F.R. § 42.4, notice is hereby given of the institution of a trial, which commences on the entry date of this decision.

IPR2020-01608
Patent 8,082,289 B2

For PETITIONER:

Brent Yamashita
Jonathan Hicks
DLA Piper LLP
brent.yamashita@dlapiper.com
jonathan.hicks@dlapiper.com
NVIDIA-ACS-IPR@us.dlapiper.com

For PATENT OWNER:

Jon W. Gurka
Ted M. Cannon
Cheryl T. Burgess
Knobbe Martens Olson and Bear, LLP
2jwg@knobbe.com
2tmc@knobbe.com
2ctb@knobbe.com
BoxZTANNL.017LP@knobbe.com

Exhibit 3

Trials@uspto.gov
571-272-7822

Paper 9
Date: May 5, 2021

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

NVIDIA CORP.,
Petitioner,

v.

ADVANCED CLUSTER SYSTEMS, INC.,
Patent Owner.

IPR2021-00075
Patent 8,140,612 B2

Before KARL D. EASTHOM, ARTHUR M. PESLAK, and
SEAN P. O'HANLON, *Administrative Patent Judges*.

EASTHOM, *Administrative Patent Judge*.

DECISION
Granting Institution of *Inter Partes* Review
35 U.S.C. § 314

IPR2021-00075
Patent 8,140,612 B2

NVIDIA Corp. (“Petitioner”) filed a Petition (Paper 1, “Pet.”) requesting an *inter partes* review of claims 1, 4–8, 10, 12, 13, 15, and 29 (the “challenged claims”) of U.S. Patent No. 8,140,612 B2 (Ex. 1001, the “’612 patent”). Advanced Cluster Systems, Inc. (“Patent Owner”) filed a Preliminary Response (Paper 5, “Prelim. Resp.”). Pursuant to the Board’s authorization (Paper 6), the parties filed additional briefing. Paper 7 (“Pet. Reply”); Paper 8 (“PO Sur-reply”).

The Board has authority to determine whether to institute an *inter partes* review (“IPR”). See 35 U.S.C. § 314(b); 37 C.F.R. § 42.4(a) (2020). Under 35 U.S.C. § 314(a), authorization of an *inter partes* review requires the information in the Petition and the Preliminary Response to “show[] that there is a reasonable likelihood that the petitioner would prevail with respect to at least 1 of the claims challenged in the petition.” For the reasons that follow, we institute an *inter partes* review as to the challenged claims of the ’612 patent on all grounds of unpatentability presented.

I. BACKGROUND

A. *Real Parties-in-Interest*

Petitioner identifies itself as the real party-in-interest. Pet. 2. Patent Owner identifies itself as the real party-in-interest. Paper 3, 1.

B. *Related Proceedings*

The parties indicate that the ’612 patent is the subject of *Advanced Cluster Systems, Inc. v. NVIDIA Corp.*, No. 19-cv-02032 (D. Del. filed Oct. 28, 2019). Pet. 3; Paper 3, 1. The parties collectively identify the following *inter partes* review petitions filed against patents related to the ’612 patent: IPR2020-01608, IPR2021-00019, IPR2021-00020, and IPR2021-00108. Pet. 3;

IPR2021-00075
Patent 8,140,612 B2

Paper 3, 1.

C. The '612 patent

The '612 patent, titled “Cluster Computing Support for Application Programs,” “relates to the field of cluster computing generally and to systems and methods for adding cluster computing functionality to a computer program, in particular.” Ex. 1001, code (54), 1:17–20.

Embodiments of the '612 patent include enabling a software package to benefit from a plurality of nodes in a cluster. *Id.* at 2:13–17. For example, “[o]ne embodiment adapts a software module designed to run on a single node, such as, for example, the Mathematica kernel, to support cluster computing, even when the software module is not designed to provide such support.” *Id.* at 2:24–27. “One embodiment provides parallelization for an application program, even if no access to the program's source code is available.” *Id.* at 2:28–30. “One embodiment provides access to [] high-performance computing through a Mathematica Front End, a command line interface, one or more high-level commands, or a programming language such as C or FORTRAN.” *Id.* at 2:20–23.

IPR2021-00075
 Patent 8,140,612 B2

Figure 1 of the '612 patent follows:

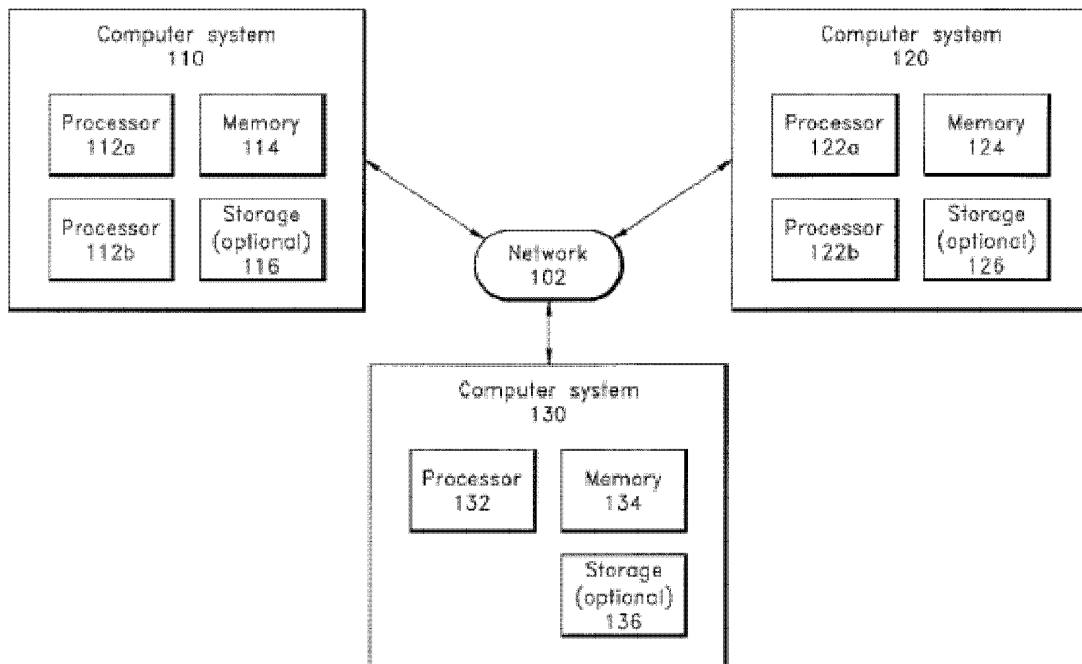


FIG. 1

Figure 1 “is a block diagram of an embodiment of a computer cluster 100 wherein computer systems 110, 120, 130 communicate with one another via a communications network 102.” Ex. 1001, 4:63–66. Each computer system includes at least one processor 112a, 112b, 122a, 122b, 132, memory 114, 124, 134, and, optionally, storage 116, 126, 136. *See id.* at 4:66–5:6. Each processor includes an independent processing core, or “node,” that is capable of single-threaded execution. *See id.* at 4:45–48, 5:6–11.

IPR2021-00075
Patent 8,140,612 B2

Figure 2 of the '612 patent, showing relationships among software modules in computer cluster 100 (Ex. 1001, 5:16–18), follows:

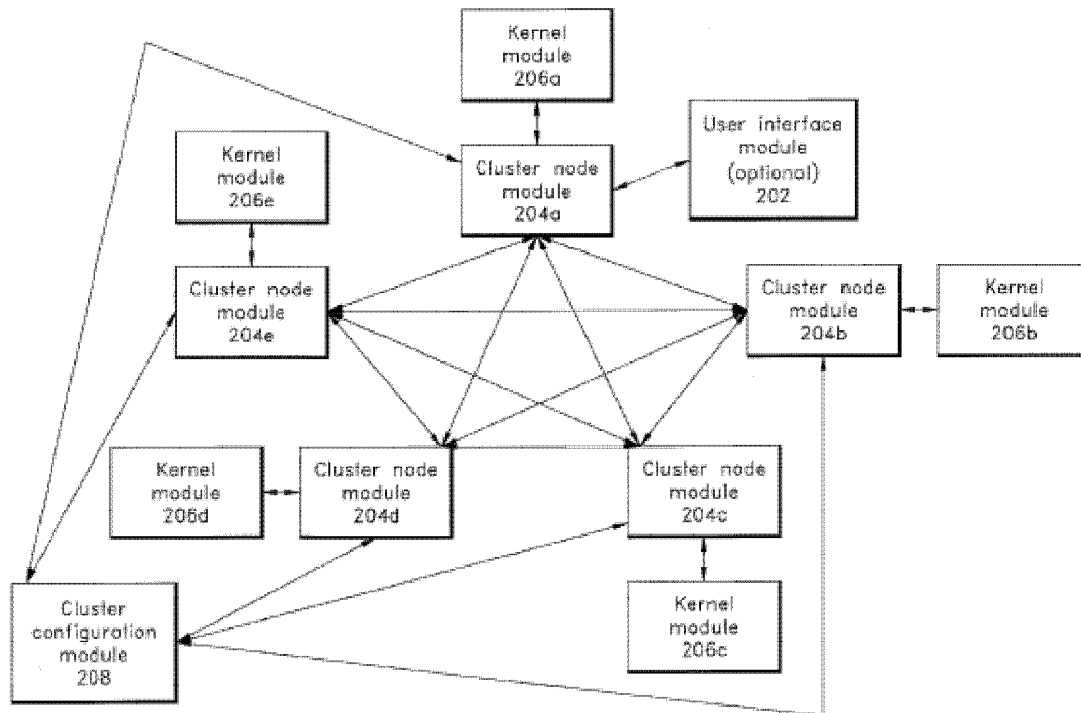


FIG. 2

Figure 2 above illustrates user interface module 202 in communication with cluster node module 204a, which in turn, is in communication with its kernel module 206a, cluster configuration module 208, and other cluster node modules 204b, 204c, 204d, and 204e, with each cluster node module respectively in communication with kernel modules 206b, 206c, 206d, and 206e. *See* Ex. 1001, 5:34–36, 5:40–45.

In the embodiment of Figure 2, each kernel module communicates with a single cluster node module. *See* Ex. 1001, 5:32–37. Cluster node modules are software modules that “can include” at least a portion of the message-passing interface (MPI) application programming interface (API) to interact with an application, such as Mathematica. *See id.* at 11:36–51. In

IPR2021-00075
 Patent 8,140,612 B2

the Figure 2 embodiment, each cluster node module communicates with each of the other cluster node modules. *Id.* at 5:40–47. Also, as indicated above, one of the cluster node modules (module 204a) communicates with user interface module 202. *Id.* at 11:6–15. That cluster node module receives commands from the user interface and submits the commands to all of the other cluster node modules. *Id.* at 23:20–24, 24:37–46. Each cluster node module communicates the command to its respective kernel module, such as for example, Mathematica kernels. *Id.* at 11:46–51, 23:24–26, 24:46–48. Each kernel module processes the command and returns a result to its respective cluster node module. *Id.* at 23:27–31, 24:61–65. The cluster node module can report the result to the other cluster node modules. *Id.* at 23:31–34, 24:65–25:1. This “interact[ion] on a pair-wise or collective basis” allows code running within multiple, simultaneously running kernel modules to perform calculations, processing, or other work on a larger scale and faster than one kernel acting alone. *Id.* at 23:55–63.

An “interpreter,” “sometimes called a kernel,” “executes instructions provided to the program by a user, a script, or another source” and “can manage at least some hardware resources of a computer system and/or can manage communications between those resources and software (for example, the provided instructions, which can include a high-level programming language).” Ex. 1001, 1:38–44.

D. Illustrative Claim 1

The Petition challenges claims 1, 4–8, 10, 12, 13, 15, and 29. Pet. 1. Claim 1, the sole independent claim, illustrates the challenged claims at issue:

IPR2021-00075

Patent 8,140,612 B2

1. [a.] A computer cluster comprising:

[b] [i.] a plurality of nodes, wherein each node is configured to access a computer-readable medium comprising [ii.] program code for a single-node kernel module configured to interpret user instructions;

[c.] [i.] a plurality of cluster node modules, wherein [ii] each cluster node module is stored in a computer-readable medium and [iii.] configured to communicate with a single-node kernel and with one or more other cluster node modules, [iv.] to accept instructions from a user interface or a script, and to interpret at least some of the user instructions such that the plurality of cluster node modules communicate with one another in order to act as a cluster, and

[d.] a communications network to connect the nodes;

[e.] wherein the plurality of cluster node modules are configured to cooperate to interpret, translate, or interpret and translate commands for execution by a plurality of single-node kernel modules, and

[f.] wherein at least one of the plurality of cluster node modules returns a result to the user interface.

Ex. 1001, 28:23–43 (bracketed information added by Board to conform to Petitioner’s nomenclature).

E. The Asserted Challenges to Patentability

The Petition relies on the following references:

Wolfgang Schreiner et al., *Distributed Maple: Parallel Computer Algebra in Networked Environments*, 35 Journal of Symbolic Computation 305 (2003) (Ex. 1008) (“Schreiner1”);

Wolfgang Schreiner, *Distributed Maple – User and Reference Manual (V 1.1.12)* (2001) (Ex. 1009) (“Schreiner2”);

IPR2021-00075
Patent 8,140,612 B2

Károly Bósa and Wolfgang Schreiner, *Task Logging, Rescheduling and Peer Checking in Distributed Maple* (2002) (Ex. 1010) (“Schreiner3”);

K. M. Heal et al., *Maple V Learning Guide* (J. S. Devitt ed., 1998) (Ex. 1011) (“Maple Guide”);

“Distributed Maple Code” references (Pet. xii):

Dist.Maple5: “[S]ource code for Distributed Maple” (Ex. 1012);

Maple Function Source Code: “Source code for parallel versions of Maple functions in Distributed Maple from the ‘distsoft’ directory” (Ex. 1013);

CASA Function Source Code: “Source code for parallel versions of CASA functions in Distributed Maple from the ‘distsoft’ directory” (Ex. 1014);

Install1 File: “‘Install’ file for Distributed Maple” (Ex. 1015);

ReadMe1 File: “‘ReadMe’ file for Distributed Maple” (Ex. 1016);

Install2 File: “‘Install’ file for source code in ‘distsoft’ directory” (Ex. 1017);

ReadMe2 File: “‘ReadMe’ file for source code in ‘distsoft’ directory” (Ex. 1018);

H.M. Deitel, P.J. Deitel, D.R. Choffnes, *Operating Systems* (3rd Ed. 2004) (Ex. 1019) (“Deitel”);

Yu Chen, Qian Fang, Zhihui Du, Sanli Li, “TH-MPI: OS Kernel Integrated Fault Tolerant MPI,” *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, Proc. 8th European PVM/MPI Users’ Group Meeting Santorini/Thera, Greece, Sept. 23–26, 2001 (“Chen”) (Ex. 1020).

Petitioner refers to Exhibits 1008–1010 and 1012–1018 collectively as “Distributed Maple Publications.” Pet. 7–8.

IPR2021-00075
Patent 8,140,612 B2

Petitioner asserts the following grounds of unpatentability:

Claim(s) Challenged	35 U.S.C. §	Reference(s)/Basis
1, 6, 10, 12, 13, 15, 29	103(a) ¹	Schreiner1, Schreiner2, Schreiner3, Maple Guide, Distributed Maple Code
4, 5, 7, 8	103(a)	Schreiner1, Schreiner2, Schreiner3, Maple Guide, Distributed Maple Code, Deitel
4, 5, 7, 8	103(a)	Schreiner1, Schreiner2, Schreiner3, Maple Guide, Distributed Maple Code, Deitel, Chen

See Pet. 10, 59, 72. Petitioner submits declarations of Henry Tufo, Ph.D. (Ex. 1005) and Wolfgang Schreiner, Ph.D. (Ex. 1006) in support of its contentions. Patent Owner submits declarations of Jaswinder Pal Singh, Ph.D. (Ex. 2001), Vineer Bhansali, Ph.D. (Ex. 2007), John Bancroft (Ex. 2008), and Dean Dager, Ph.D. (Ex. 2033), in support of its Preliminary Response.

II. ANALYSIS

Petitioner challenges claims 1, 4–8, 10, 12, 13, 15, and 29 as obvious based on the grounds listed above. Pet. 1. Patent Owner disagrees. Prelim. Resp. 1–8.

¹ The filing date of the application resulting in the '612 patent precedes the effective date of Leahy-Smith America Invents Act (“AIA”), Pub. L. No. 112–29, 125 Stat. 284 (2011). Thus, reference is to the pre-AIA version of section 103.

IPR2021-00075
 Patent 8,140,612 B2

A. Legal Standards

35 U.S.C. § 103(a) renders a claim unpatentable if the differences between the claimed subject matter and the prior art are such that the subject matter, as a whole, would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. *See KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 406 (2007).

Tribunals resolve obviousness based on underlying factual determinations, including (1) the scope and content of the prior art; (2) any differences between the claimed subject matter and the prior art; (3) the level of skill in the art; and (4) where in evidence, so-called secondary considerations. *See Graham v. John Deere Co.*, 383 U.S. 1, 17–18 (1966). Prior art references must be “considered together with the knowledge of one of ordinary skill in the pertinent art.” *In re Paulsen*, 30 F.3d 1475, 1480 (Fed. Cir. 1994) (citing *In re Samour*, 571 F.2d 559, 562 (CCPA 1978)).

B. Level of Ordinary Skill in the Art

Petitioner relies on Dr. Tufo, who testifies that a person having ordinary skill in the art at the time of the invention (“POSITA”) would have had “a Bachelor’s degree in computer science, electrical engineering, or an equivalent field, and two years of academic or industry experience in parallel and/or distributed computing.” Ex. 1005 ¶ 40; Pet. 13 (citing Ex. 1005 ¶¶ 38–41).

On one hand, the Preliminary Response does not present a proposed level of ordinary skill at this preliminary stage. On the other hand, Dr. Singh, Patent Owner’s declarant, disagrees with Dr. Tofu’s proposed level of

IPR2021-00075
 Patent 8,140,612 B2

ordinary skill. Ex. 2001 ¶¶ 33–34.² The Preliminary Response does not cite Dr. Singh’s proposal. For purposes of this Decision on Institution, we adopt Petitioner’s proposed level of ordinary skill in the art, which comports with the teachings of the ’612 patent and the asserted prior art.

C. Claim Construction

In an *inter partes* review, the Board construes each claim “in accordance with the ordinary and customary meaning of such claim as understood by one of ordinary skill in the art and the prosecution history pertaining to the patent.” 37 C.F.R. § 42.100(b) (2020). Under the same standard applied by district courts, claim terms have their plain and ordinary meaning as would have been understood by a person of ordinary skill in the art at the time of the invention and in the context of the entire patent disclosure. *Phillips v. AWH Corp.*, 415 F.3d 1303, 1313 (Fed. Cir. 2005) (en banc). “There are only two exceptions to this general rule: 1) when a patentee sets out a definition and acts as his own lexicographer, or 2) when the patentee disavows the full scope of a claim term either in the specification or during prosecution.” *Thorner v. Sony Comput. Entm’t Am. LLC*, 669 F.3d 1362, 1365 (Fed. Cir. 2012).

² Dr. Singh contends that “‘parallel computing’ is a category that includes ‘cluster computing’ but ‘distributed computing’ is different from both ‘cluster computing’ and ‘parallel computing.’” *Id.* ¶ 33. Accordingly, Dr. Singh testifies that a “POSITA would have had a Bachelor’s degree in computer science, electrical engineering, or an equivalent field, and two years of academic or industry experience in **cluster computing**.” *Id.* ¶ 34. However, Dr. Singh states that “my conclusions would not change even assuming that Dr. Tufo’s identification of the level of ordinary skill in the art were correct.” *Id.* For purposes of institution, Dr. Singh’s testimony reveals that any differences in the proposed level of ordinary skill raised by Dr. Singh are not material.

IPR2021-00075
Patent 8,140,612 B2

Claim 1 recites the following “cluster node module” limitation, which raises the sole claim construction under dispute here:

a plurality of cluster node modules, wherein each cluster node module is stored in a computer readable medium and configured to communicate with a single-node kernel and with one or more other cluster node modules, to accept instructions from a user interface or a script, and to interpret at least some of the user instructions such that the plurality of cluster node modules communicate with one another in order to act as a cluster.

Generally, “Petitioner believes that no express construction of any term is needed to resolve the challenges herein but reserves the right to present express constructions in response to any argument by P[atent] O[wner].” Pet. 13.³ Notwithstanding Petitioner’s statement, Patent Owner argues that “Petitioner . . . failed to identify ‘[h]ow the challenged claim is to be construed’ and ‘[h]ow the construed claim is unpatentable,’” as required by 37 C.F.R. §§ 42.104(b)(3) & 42.104(b)(4).” Prelim. Resp. 50. To the contrary, Petitioner sufficiently identifies how it reads the claims onto Schreiner1’s system, as discussed below. *See infra* § II.D.6. Also, it is

³ Neither party argues that term “module” as recited in claim 1 (i.e., “cluster node module” or “kernel module”) is a means-plus-function term even though “‘module’ is a well-known nonce word that can operate as a substitute for ‘means’ in the context of § 112, para. 6.” *See Williamson v. Citrix Online, LLC*, 792 F.3d 1339, 1350 (Fed. Cir. 2015) (en banc) (“use of the word ‘means’ creates a presumption that [35 U.S.C.] § 112, ¶ 6 applies”); *see* 35 U.S.C.] § 112, ¶ 6 (“An element in a claim for a combination may be expressed as a means . . . for performing a specified function without the recital of structure, material, or acts in support thereof, and such claim shall be construed to cover the corresponding structure, material, or acts described in the specification and equivalents thereof.”). Because neither party argues that “module” is a means-plus-function term, we decline to construe it as such for institution purposes.

IPR2021-00075
 Patent 8,140,612 B2

evident from Patent Owner’s claim construction arguments (addressed in this section and further below (*id.*)) that Patent Owner disagrees with, and therefore understands, the central claim construction dispute underlying Petitioner’s implicit claim construction of “cluster node module.”

Specifically, Patent Owner presents two separate requirements for the “cluster node module” as recited in claim 1. First, Patent Owner contends “the claims require the cluster node modules to collectively accept instructions from the user interface *without the instructions first passing through any kernel*. After accepting instructions, the cluster node modules may optionally send the instructions to kernels.” Prelim. Resp. 1 (emphasis added).

Second, Patent Owner contends

the Board should construe “cluster node module” to mean “a module that cooperates with other cluster node modules to establish intercommunication among nodes in a computer cluster and to exchange messages such that each node can communicate tasks and data with other nodes *without the tasks and data being required to go through a central server or master node*.”

Prelim. Resp. 15–16 (emphasis added).

Both of Patent Owner’s contentions seek to import negative limitations into claim 1. Further, regarding the first requirement, Patent Owner argues claim 1 “excludes communication of instructions from the user interface to a kernel, which then forwards the instructions to a cluster node module.” Prelim. Resp. 9. Patent Owner sets forth its argument in the following diagram:

IPR2021-00075
 Patent 8,140,612 B2

Correct order:	User Interface → Cluster Node Module → (Kernel)
Incorrect order:	User Interface → Kernel → Cluster Node Module

The diagram above represents Patent Owner’s claim construction wherein Patent Owner labels as an “[i]ncorrect order” a construction of claim 1 that allows a kernel to pass an instruction from a user interface to a cluster node module. *See* Prelim. Resp. 9 (citing Ex. 2001 ¶ 41).

Contrary to Patent Owner’s arguments, the plain language of claim 1 does not preclude instructions from passing from a user interface via an intervening kernel to a cluster node module and does not require any of the other negative limitations argued. Rather, claim 1 recites “each cluster node module is . . . configured to communicate with a single-node kernel and with one or more other cluster node modules, to accept instructions from a user interface or a script, and to interpret at least some of the user instructions such that the plurality of cluster node modules communicate with one another in order to act as a cluster.” This language requires “each cluster node module . . . to communicate with a single-node kernel” without specifying the kernel’s position relative to a user interface and cluster node module, and also without specifying anything about a central server or master mode. The language also requires “each cluster node module . . . to accept instructions from a user interface or a script, and to interpret at least some of the user instructions.” As indicated, this language does not prevent the instructions from passing through a kernel, central server, or master node situated between the user interface or script and each cluster node module. The language of the claim 1 also does not specify the origin of “a script” or

IPR2021-00075
 Patent 8,140,612 B2

preclude the script from coming from a kernel, central server, or master mode.

Contrary to Patent Owner’s other arguments, the specification does not limit the claims in the manner argued. *See* Prelim. Resp. 10–11 (citing Ex. 1001, 21:24–26, Fig. 2). Patent Owner relies on Figure 2 and other selected passages to incorporate limitations into claim 1 (*see id.*), but the specification specifically states that “[t]he drawings and the associated descriptions are provided to illustrate embodiments and *not to limit the scope of the disclosure.*” Ex. 1001, 3:63–65 (emphasis added). Moreover, following guidance from the Federal Circuit, generally “[w]e do not read limitations from the embodiments in the specification into the claims.” *Hill-Rom Servs., Inc. v. Stryker Corp.*, 755 F.3d 1367, 1371 (Fed. Cir. 2014) (citing *Liebel-Flarsheim Co. v. Medrad, Inc.*, 358 F.3d 898, 904 (Fed. Cir. 2004)).

Similarly, the specification describes several embodiments under a “SUMMARY” of the invention section in general terms “[w]ithout limiting the scope of the invention.” *See* Ex. 1001, 2:7–8. One passage mimics the broad language of claim 1, which plainly does not provide the negative limitations argued by Patent Owner:

Each cluster node module is configured to communicate with a single node kernel and with one or more other cluster node modules, to accept instructions from the user interface, and to interpret at least some of the user instructions such that the plurality of cluster node modules communicate with one another in order to act as a cluster.

Ex. 1001, 3:22–28.

This generic passage allows each cluster node module “to accept instructions from the user interface,” without requiring it to accept them

IPR2021-00075
 Patent 8,140,612 B2

directly from the user interface. The passage says nothing about precluding passage of a message through a master node, central server, or kernel.

Patent Owner also relies on Figure 2 as “show[ing] that the user interface module 202 is connected to the cluster node module 204a *only* and the kernel module 206a is connected to the cluster node module 204a *only*.” Prelim. Resp. 11. Patent Owner similarly contends that “[i]n embodiments that include cluster node modules, there is no connection between the user interface module 202 and the kernel module 206a or any of the other kernel modules.” *Id.* at 11–12.

As noted above, however, embodiments in the specification generally do not limit the claim terms. *See Hill-Rom Servs.*, 755 F.3d at 1371. Also, the specification contradicts Patent Owner’s arguments on this preliminary record. For example, it states “[a] kernel module 206 typically includes program code for interpreting high-level code, commands, and/or instructions *supplied by a user or a script* into low-level code, such as, for example, machine language or assembly language.” Ex. 1001, 21:33–36 (emphasis added). Similarly, the specification specifically states that “[t]he operating system or its components *can connect the front end 202 and kernels 206 to one another in the same manner as a cluster node module 204* would or by a variation of one of the techniques described previously.” Ex. 1001, 24:22–26 (emphasis added). In other words, contrary to Patent Owner’s arguments, on this preliminary record, the specification supports connecting a user interface directly to any kernel.

In line with the above teachings, the specification also describes user interface module 202 communicating with kernel module 202 for “some embodiments” as follows:

IPR2021-00075

Patent 8,140,612 B2

In *some embodiments*, computer cluster 100 includes a user interface module 202, such as, for example a Mathematica Front End or a command line interface, *that includes program code for a kernel module 206 to provide graphical output, accept graphical input, and provide other methods of user communication that a graphical user interface or a command-line interface provides. To support a user interface module 202, the behavior of a cluster node module 204a is altered in some embodiments.* Rather than sending output to and accepting input from the user directly, the user interface module 202 activates the cluster node module 204a to which it is connected and specifies parameters to form a connection, such as a MathLink connection, between the cluster node module 204a and the user interface module 202. The user interface module's activation of the cluster node module 204a can initiate the execution of instructions to activate the remaining cluster node modules 204b–e on the cluster and to complete the sequence to start all kernel modules 206a–e on the cluster. *Packets from the user interface module 202, normally intended for a kernel module 206a, are accepted by the cluster node module 204a as a user command.* Output from the kernel module 206a associated with the cluster node module 204a can be forwarded back to the user interface module 202 for display to a user. Any of the cluster node modules 204a–e can be configured to communicate with a user interface module 202.

Ex. 1001, 21:6–31 (emphasis added).

Patent Owner argues that portions of this passage support its construction. Prelim. Resp. 10 (citing Ex. 1001, 21:24–26). However, in context, the passage, read in its entirety, does not support Patent Owner's narrow claim construction on this preliminary record. Rather, the first emphasized portion as quoted above explicitly describes communication between user interface module 202 and kernel module 206. It also describes “alter[ing]” “the *behavior* of cluster node module 204a . . . [but only] *in some embodiments*” so that the cluster node module “can initiate the

IPR2021-00075
Patent 8,140,612 B2

execution of instructions to activate the remaining cluster node modules 204b–e on the cluster and to complete the sequence to start all kernel modules 206a–e on the cluster.” *See id.* (alterations and emphasis by Board). In other words, to the extent altering this “behavior” involving cluster node modules somehow limits normal behavior with respect to normal kernel communications, it only occurs for “some embodiments”—i.e., a subset of “some embodiments” introduced at the beginning of the passage.

In addition, the passage verifies that packets *normally* pass to kernel module 206a (at least for some contemplated embodiments). *See* Ex. 1001, 21:6–31. Therefore, it is only in a subset of the embodiments that packet messages pass from user interface module 202 first, then through cluster node module 204, and finally to kernel module 206a. *See id.* Accordingly, nothing in this passage limits the claims to a direct connection between a user interface and a cluster node module, by precluding an intervening kernel, master node, or server. In addition, Patent Owner’s claim construction attempts to allow some forms of indirect communication between cluster nodes, by only attempting to preclude an intervening “*central* server” or “*master* mode,” leaving the claim construction open to interpretation without requisite support from the specification. Similarly, Patent Owner explicitly states that its claim construction does not require a direct connection between a user interface and a cluster node module, because it allows for intervening “devices” or “components.” *See* Prelim. Resp. 14 n.4 (“To be clear, this construction does not require instructions to be transmitted directly from the user interface to a cluster node module

IPR2021-00075
Patent 8,140,612 B2

without passing through other devices, such as routers, switches, or other components.”).

In addition, cluster node module 204a appears to act as a “master node” or “central server” when it is connected to the user interface module, because messages from other kernels (nodes) must pass through cluster node module 204a on their way to the kernel associated with cluster node module 204a and/or to the user interface node.⁴ *See, e.g.*, Fig. 2, Ex. 1001, 6:17–21 (“Results of evaluations performed by kernel modules 206a-e are communicated back to the first cluster node module 204a via the cluster node modules 204a-e, which communicates them to the user interface module 208.”), 11:33–36 (“In one embodiment, the cluster node modules 204a-e provide a way for many kernel modules 206a-e such as, for example, Mathematica kernels, running on a computer cluster 100 to communicate with one another.”), 22:48–50 (“The cluster node module creates an illusion that a kernel module is communicating directly with the other kernel modules.”). Cluster node module 204a also acts as a “central server,” because it instigates connections to the remaining cluster node modules, according to the column 21 passage discussed and reproduced above. It also controls the other cluster node modules in a “procedure to shut down the system.” *See id.* at 24:37–53.

The specification also indicates that a load balancing embodiment includes a “root processor” that assigns tasks to each of the cluster nodes.

⁴ According to the specification, “[t]he term ‘node’ refers to a processing unit or subunit that is capable of single-threaded execution of code.” Ex. 1001, 4:46–48. The specification also describes “computers, microprocessors, and/or processor cores (‘nodes’).” *Id.* at 1:24–25.

IPR2021-00075
Patent 8,140,612 B2

See Ex. 1001, 19:60–20:4. On this preliminary record, this further shows that the claims do not require a cluster node module “to exchange messages such that each node can communicate tasks and data with other nodes *without the tasks and data being required to go through a central server or master node*” as asserted by Patent Owner.

Finally, on this preliminary record, the prosecution history also does not support the negative limitations argued by Patent Owner. *See* Ex. 1002, 1 (“Applicant hereby rescinds and retracts such disclaimer” arising out of “any prior amendments or characterizations of the scope of any claim or referenced art.”).

Based on the foregoing discussion, requisite disclaimer, disavowal, or lexicography does not appear to exist on this preliminary record to import Patent Owner’s proposed negative limitations into the plain language of independent claim 1. *See Omega Eng’g, Inc. v. Raytek Corp.*, 334 F.3d 1314, 1321–23 (Fed. Cir. 2003) (holding that a claim limitation to “strike the periphery . . . for visibly outlining” an energy zone with a laser does not justify a negative limitation of “not striking the center or interior portion” of the energy zone absent an “express disclaimer or independent lexicography in the written description that would justify adding that negative limitation”).

No other terms require an express construction. Only those terms that are in controversy need be construed, and only to the extent necessary to resolve the controversy. *Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co.*, 868 F.3d 1013, 1017 (Fed. Cir. 2017) (citing *Vivid Techs., Inc. v. Am. Sci. & Eng’g, Inc.*, 200 F.3d 795, 803 (Fed. Cir. 1999)).

IPR2021-00075
Patent 8,140,612 B2

D. Alleged Obviousness of Claims 1, 6, 10, 12, 13, 15, and 29

Petitioner relies on the combined teachings of Schreiner1, Schreiner2, Schreiner3, Maple Guide, and Distributed Maple Code to allege obviousness of claims 1, 6, 10, 12, 13, 15, and 29. *See* Pet. 14–59.

1. Schreiner1 (Ex. 1008)

Schreiner1, entitled “Distributed Maple: parallel computer algebra in networked environments,” bears a copyright date of 2003. Ex. 1008, 3. Schreiner1 is a journal article authored by Dr. Schreiner, Christian Mittermaier, and Karoly Bosa. *Id.* at 3. Schreiner1 “gives a comprehensive overview on the design and the use of ‘Distributed Maple,’ an environment for parallel computer algebra on multiprocessors and heterogeneous computer clusters.” *Id.* at 3. Schreiner1 explains that Distributed Maple was developed on the basis of the computer algebra system Maple (*id.*) and that Distributed Maple is built on top of the Maple kernel and does not require any kernel extensions. *Id.* at 4. Schreiner1 states that Distributed Maple is “so portable that applications can be executed in many different environments” and “so general that it can be applied to schedule tasks of other computer algebra systems (e.g., Mathematica).” *Id.* Schreiner1 describes Distributed Maple as providing “a programming model which is based on functional/logic/dataflow parallelism” that “allows the creation of a large number of implicitly scheduled tasks with automatic resolution of data dependencies and of globally shared data structures with implicit synchronization.” *Id.*

Schreiner1 describes using Distributed Maple “to develop the first parallel versions for a number of non-trivial applications from algebraic geometry (parallel curve and surface plotting and parallel neighborhood

IPR2021-00075
 Patent 8,140,612 B2

analysis).” Ex. 1008, 5. Schreiner1 discloses that “[t]he user interacts with Distributed Maple via a conventional Maple frontend (text or graphical).” *Id.* at 7. Schreiner1 explains that “[t]he core of Distributed Maple is a scheduler program which is completely independent and even unaware of Maple” and “can in fact embed and schedule tasks from any kind of computation kernels that implement a specific communication protocol.” *Id.* at 8.

Schreiner1 discloses that a Distributed Maple session comprises two main components: a scheduler and a Maple interface. Ex. 1008, 8. Figure 1 of Schreiner1, reproduced below, depicts these components and corresponding software architecture for a Distributed Maple session. *Id.* at 9.

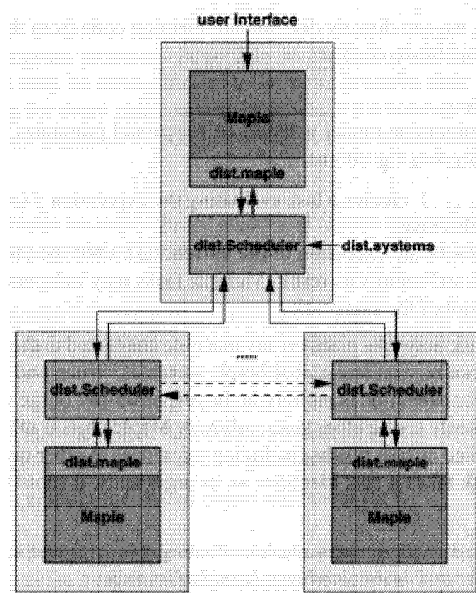


Fig. 1. Software architecture.

As shown in Figure 1, a Distributed Maple session “comprises a set of nodes each of which holds a pair of processes: a *kernel* and a *scheduler*.” Ex. 1008, 17. “Initially, a single task runs on the root kernel; this task may

IPR2021-00075
Patent 8,140,612 B2

subsequently create new tasks which are distributed via the schedulers to other kernels and may in turn create new tasks.” *Id.* With reference to Figure 1, Schreiner1 explains that “every scheduler instance accepts tasks from the attached computation kernel and schedules these tasks among all machines connected to the session.” *Id.* at 9. Schreiner1 further explains that “[t]he Maple kernel is a single-threaded process which communicates by a simple communication protocol with the scheduler on the same node” and “[a]ll capabilities for parallel and distributed program execution are embedded in this scheduler.” *Id.* at 12.

In general, “[t]he system embeds kernels of the computer algebra system Maple as computational engines into a networked coordination layer implemented in the programming language Java.” Ex. 1008, 3 (Abstract). Maple is a computer algebra system similar to that of Mathematica. *See id.* at 4. The Distributed Maple system “connects external computation kernels on various machines and schedules concurrent tasks for execution on them.” *Id.*

Using “a comparatively high-level programming model, one may write parallel Maple programs that show good speedups in medium-scaled environments.” Ex. 1008, 3 (Abstract). Schreiner1 states “[b]oth the Distributed Maple system itself and the library of parallel version of . . . Maple algorithms are in stable versions freely available under the GUN Library General Public License at <http://www.risc.uni-linz.ac.at/software/distmaple>.” *Id.* at 5.

2. *Schreiner2 (Ex. 1009)*

Schreiner2, titled “Distributed Maple – User and Reference Manual (V 1.1.12),” by Dr. Schreiner, published on July 6, 2001. Ex. 1009, 1.

IPR2021-00075
Patent 8,140,612 B2

Schreiner2 describes the same Maple and Distributed Maple system as Schreiner1 in further detail, including programming and system details. *See, e.g., id.* at 1, 4, 13–19. More particularly, Schreiner2 “describes the use of a system for writing distributed Maple applications and sketches its implementation.” *Id.* at 4. Schreiner3 states that its “goal is to provide an environment that makes it easy to implement parallel algorithms in Maple and that can be easily installed in any environment in order to facilitate the distribution of these implementations.” *Id.* at 4. As in Schreiner1, Schreiner2 states “[t]he system . . . can be freely downloaded from <http://www.risc.uni-linz.ac.at/software/distmaple>.” *Id.* at 1.

3. *Schreiner3 (Ex. 1010)*

Schreiner3, titled “Task Logging, Rescheduling and Peer Checking in Distributed Maple,” by Dr. Schreiner and others, published on March 18, 2002. Exhibit 1010, 1. Schreiner3 describes the same Maple and Distributed Maple system as Schreiner1 in further detail, including system details and fault analysis. *See id.* at 1–14. Schreiner3 cites Schreiner2. *Id.* at 44 (also citing <http://www.risc.uni-linz.ac.at/software/distmaple>).

IPR2021-00075
 Patent 8,140,612 B2

Figure 1 of Schreiner3 illustrates an Execution Model of the Distributed Maple system:

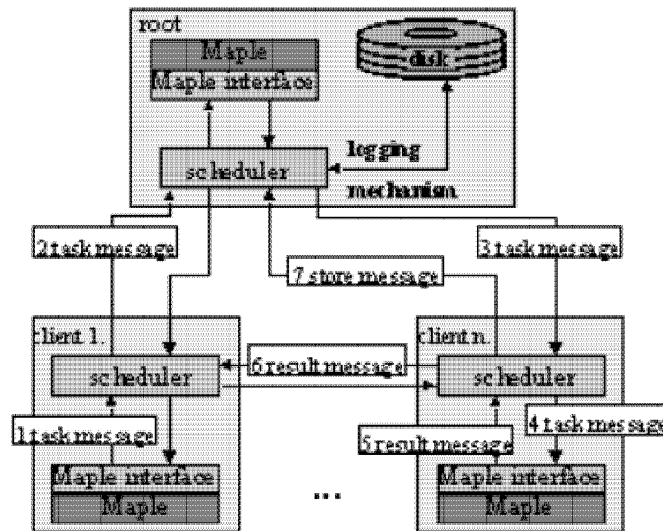


Figure 1: Execution Model

Figure 1 depicts the passing of messages between nodes in a Distributed Maple system, and a logging mechanism in the root node. Ex. 1010, 5.

Schreiner3 explains that “[t]he logging mechanism in Distributed Maple is a fault tolerance mechanism for saving the results of intermediate tasks and the values of shared objects during the computation” and allows the system “to restore the results of computed tasks in a later session, if the current session crashes.” Ex. 1010, 3.

4. Maple Guide (Ex. 1011)

Maple Guide, entitled “Maple V Learning Guide, Release 5,” and published by Waterloo Maple, Inc., bears a copyright date of 1998. Ex. 1011, 5. Maple Guide explains that “Maple V is a *Symbolic Computation System* or *Computer Algebra System*” and that “[b]oth phrases refer to Maple V’s ability to manipulate information in a symbolic or

IPR2021-00075
 Patent 8,140,612 B2

algebraic manner.” *Id.* at 11. Maple Guide is an introductory guide that describes how to use Maple’s graphical interface and the Maple programming language. *Id.* at 13.

5. *Distributed Maple Code (Ex. 1012–Ex. 1018)*

Distributed Maple Code is a collection of the following computer code files embodying a distribution of Distributed Maple for installation on a computer: dist.maple5 file (Ex. 1012), source code for parallel versions of Maple functions in the distsoft directory (Ex. 1013), source code for parallel versions of CASA functions in the distsoft directory (Ex. 1014), an “Install” file for Distributed Maple (Ex. 1015), a “ReadMe” file for Distributed Maple (Ex. 1016), an “Install” file for the source code in the distsoft directory (Ex. 1017), and a “ReadMe” file for the source code in the distsoft directory (Ex. 1018).

6. *Claims 1, 6, 10, 12, 13, 15, and 29*

a. Analysis of Claim 1

Petitioner relies on the combined teachings of Schreiner1, Schreiner2, Schreiner3, Maple Guide, and Distributed Maple Code (source code), as supported by the testimony of Dr. Tufo and Dr. Schreiner, to allege obviousness of claims 1, 6, 10, 12, 13, 15, and 29. *See* Pet. 18–54.

As motivation to combine the “Distributed Maple Publications,” Petitioner contends that they share the same author, Dr. Schreiner, and all relate to the same software project, called “Distributed Maple.” Pet. 14. Petitioner essentially contends that a person of ordinary skill would have consulted the references to learn details about the system, including fault tolerances and capabilities, in order to combine desired features for running the software modules and system. *See* Pet. 14–16.

IPR2021-00075
 Patent 8,140,612 B2

Regarding the Maple Guide (not authored by Dr. Schreiner), according to Petitioner, “Schreiner1 teaches that Distributed Maple includes Maple software modules and refers readers to www.maplesoft.com, a website operated by Waterloo Maple, the company that authored and sold the Maple software, for further details.” Pet. 15 (citing Ex. 1008, 44; Ex. 1006 ¶ 49).⁵ Petitioner explains that

Waterloo Maple published the Maple Guide, and a POSITA would have been motivated to read the Maple Guide to learn more about Maple. The teaching in the Distributed Maple Publications that Distributed Maple utilized Maple, including its kernel and libraries, provides a POSITA with a strong, express motivation to combine the features described in the Distributed Maple Publications with the features of Maple, as described in the Maple Guide.

Id. at 16 (citing Ex. 1005 ¶ 47).

Petitioner also contends that “the references . . . were publicly available on the same webpage – <http://www.risc.unilinz.ac.at/software/distmaple> – which was cited by Schreiner1 and date-stamped and archived by the Internet Archive, and is submitted as Exhibits 1024 and 1025.” Pet. 15 (citing Ex. 1008, 5–6; Ex. 1009, Abstract, 4; Ex. 1005 ¶¶ 45–46; Ex. 1006 ¶¶ 24–25; Ex. 1024; Ex. 1025). As noted above, Schreiner1 states that “[b]oth the Distributed Maple system itself and the library of parallel version of . . . Maple algorithms are in stable versions freely

⁵ Describing Distributed Maple as using “a conventional Maple frontend,” Schreiner1 states that “‘Maple’ is a registered of ‘Waterloo Maple Inc.’” Ex. 1008, 7. Schreiner1 also cites <http://www.maplesoft.com> under a listing of reference sources, listing “Maple, W., Maple 6, 2001” as one such reference source. *Id.* at 44.

IPR2021-00075
Patent 8,140,612 B2

available under the GUN Library General Public License at
<http://www.risc.uni-linz.ac.at/software/distmaple>.” Ex. 1008, 5.

Petitioner maintains that regardless of the motivation as summarized above, “Schreiner1 expressly teaches nearly all of the claim limitations by itself, and further motivations to combine for specific features are detailed below in connection with particular claim limitations.” Pet. 16 (citing Ex. 1005 ¶ 48).

Claim’s 1 preamble recites “[a] computer cluster comprising.” *See* Pet. 22 (designating the preamble “a.”). Petitioner contends that even if the preamble limits the claim, “Schreiner1 discloses it.” *Id.* Petitioner relies on Schreiner1’s abstract, which states “[w]e describe the design and use of Distributed Maple, an environment for executing **parallel computer algebra programs on multiprocessors and heterogeneous clusters**.” *Id.* (Ex. 1008, Abstract). Petitioner also contends Schreiner1’s Figure 1 depicts a cluster and Schreiner1 otherwise describes “parallel operations on various clusters.” *Id.* (citing Ex. 1008, Fig. 1, 22–42; Ex. 1005 ¶ 60).

Claim 1 also recites the following limitations:

[b.] “a plurality of nodes, wherein each node is configured to access a computer-readable medium comprising program code for a single-node kernel module configured to interpret user instructions;”

[i.] “a plurality of nodes, wherein each node is configured to access a computer-readable medium”

[ii.] “comprising program code for a single-node kernel module configured to interpret user instructions”

IPR2021-00075
Patent 8,140,612 B2

To address these limitations, Petitioner annotates Schreiner1's Figure 1 as follows (Pet. 23):

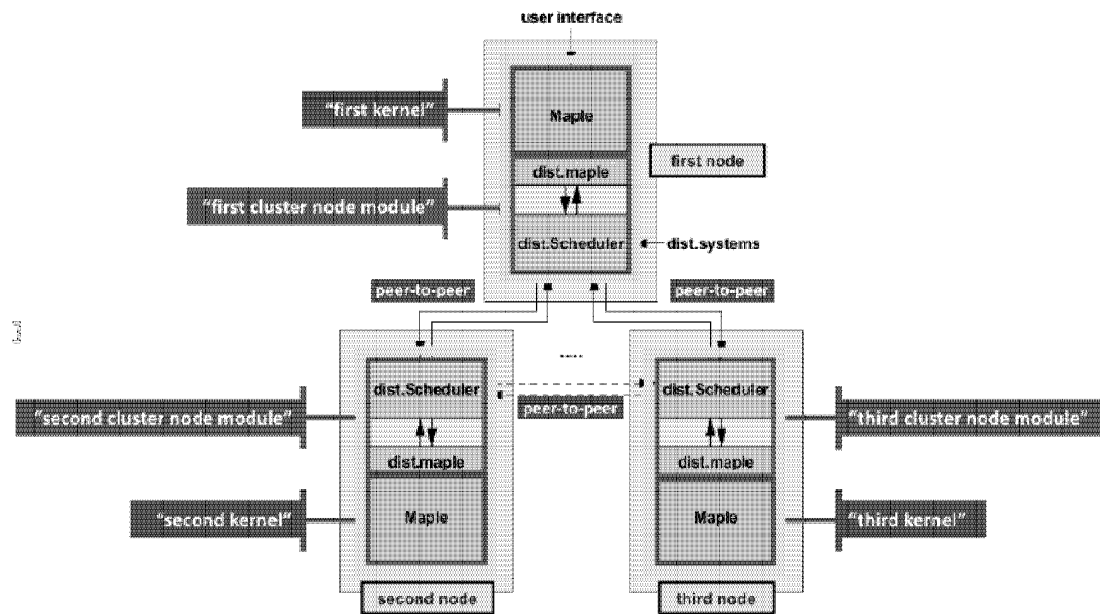


Fig. 1. Software architecture.

According to Petitioner’s annotations above in Figure 1, Figure 1 shows Maple kernels (first kernel, second kernel, and third kernel) at each node (first node, second node, and third node). Pet. 23 (citing Ex. 1008, 9; Ex. 1005 ¶ 61). Further addressing limitations b and b.i, Petitioner states that “Maple is installed in storage and loaded in memory when the program is run by one or more processors.” *Id.* (citing Ex. 1011, 5, 92). Petitioner also contends that [t]he Maple “kernel consists of highly optimized C code.” *Id.* (quoting Ex. 1011, 88). Petitioner also contends that “Distributed Maple” is a software program loaded into memory for execution by one or more processors. *Id.* at 24 (citing Ex. 1008, 5, 22–42; Ex. 1009, 9; Ex. 1015; Ex. 1016; Ex. 1005 ¶ 64). Petitioner also points to “shared memory” as disclosed in Schreiner1 as a computer-readable memory for running Maple

IPR2021-00075
 Patent 8,140,612 B2

and holding the Maple kernels and Distributed Maple code. *Id.* (citing Ex. 1008, 9, 12, 25; Ex. 1009, 9; Ex. 1005 ¶ 65).

Addressing limitation b.ii, Petitioner contends that Schreiner teaches use of Maple as a single-node kernel on each node of the Distributed Maple cluster. Pet. 25 (citing Ex. 1008, 8–9). Petitioner also contends that each Maple kernel interprets user instructions. *Id.* For example, Petitioner contends that that “Distributed Maple ‘embeds kernels of the computer algebra system Maple as computational engines’ and employs ‘a comparatively high-level programming model.’” *Id.* (quoting Ex. 1008, Abstract; citing *id.* at 30; 1005 ¶ 68.).

To further support its showing, Petitioner quotes the Maple Guide:

The kernel is the base of Maple’s system. It contains fundamental and primitive commands: *the Maple language interpreter (which converts the commands you type into machine instructions your computer processor can understand)*, algorithms for numerical calculation, and routines to display results and perform other input and output operations. . . . The Maple kernel implements the most frequently used routines for integer and rational arithmetic and simple polynomial calculations.

Pet. 25–26 (quoting Ex. 1011, 88) (emphasis by Petitioner). Petitioner also explains that “Schreiner2 teaches that high-level commands are translated by the kernel into lower-level code for execution by the processors.” *Id.* at 26 (citing Ex. 1009, 7, 15, 30). According to Petitioner, “Schreiner1 . . . discloses using Distributed Maple with a Mathematica kernel, exactly like in the embodiments of the ’612 patent.” *Id.* (citing Ex. 1008, 4; Ex. 1005 ¶ 71).

IPR2021-00075
 Patent 8,140,612 B2

Claim 1 also recites the following limitations:

[c.] [i.] a plurality of cluster node modules, wherein [ii] each cluster node module is stored in a computer-readable medium and [iii.] configured to communicate with a single-node kernel and with one or more other cluster node modules, [iv.] to accept instructions from a user interface or a script, and to interpret at least some of the user instructions such that the plurality of cluster node modules communicate with one another in order to act as a cluster, and

Generally addressing the cluster node modules and single-node kernel limitations, Petitioner annotates Schreiner1's Figure 1 (as reproduced above) to depict first, second, and third cluster node modules (blue) communicating in a peer-to-peer fashion with each other and communicating with single-node Maple kernels (red). *See* Pet. 28 (annotating Schreiner's Figure 1). Petitioner explains that Schreiner's "Figure 1 depicts two parts of Distributed Maple's [blue] first cluster node module: `dist.maple` and `dist.Scheduler`." *Id.* (citing Ex. 1008, Fig. 1; Ex. 1005 ¶ 75).

According to Petitioner, "[t]he `dist.Scheduler` and `dist.maple` modules work together [as a (blue) cluster node module] to provide communication capabilities: `dist.Scheduler` 'coordinates node interaction,' and `dist.maple` 'implements the interface between the [red] kernel [Maple] and the scheduler.'" Pet. 28 (quoting Ex. 1008, 8). To support its showing, Petitioner quotes the following passage from Schreiner1: "The Maple kernel is a single-threaded process which communicates by a simple communication protocol with the scheduler on the same node. All capabilities for parallel and distributed program execution are embedded in this scheduler." *Id.* (quoting Ex. 1008, 12).

Petitioner also quotes from Schreiner2 to describe the scheduler and Maple interface:

IPR2021-00075
 Patent 8,140,612 B2

Scheduler A scheduler program manages the node interaction and schedules tasks among nodes. This program is implemented by a Java class library with main class *dist.Scheduler* and is independent of Maple. The initial scheduler process reads all application-specific information from the configuration file *dist.systems* and may then start instances of the scheduler on other machines with which it communicates via sockets.

Maple Interface The package *dist.maple* running on each Maple kernel implements the interface between Maple and the scheduler. Communication between both components is based on pipes (named pipes for the Maple kernel connected to the user interface and standard input/output streams for the backend kernels). During a session, additional socket connections between remote scheduler instances are dynamically established on demand.

Pet. 29 (quoting Ex. 1009, 24; citing Ex. 1005 ¶ 76).

Addressing limitation c.ii, Petitioner relies on its showing above with respect to limitation b.i, contending that the “[t]he Distributed Maple software was installed in a storage device and loaded into memory during operation.” Pet. 30 (citing Ex. 1005 ¶ 78).

Addressing limitation c.iii, Petitioner explains that the scheduler sends some of the Distributed Maple commands to all kernels. Pet. 31 (citing Ex. 1008, 9–10, 14; Ex. 1009, 15). As an example, Petitioner quotes Schreiner¹ as follows: “After a session has been established, every scheduler instance accepts tasks from the attached computation kernel and schedules these tasks among all machines connected to the session.” Pet. 31 (quoting Ex. 1008, 9). “All remote schedulers send new tasks to the root node scheduler which distributes them among all machines.” *Id.* (quoting Ex. 1008, 14). “[T]he scheduler accepts tasks from the Maple process and schedules these tasks among any node connected to the session.” *Id.* (quoting Ex. 1008, 14).

IPR2021-00075
 Patent 8,140,612 B2

Relying on its annotated version of Schreiner's Figure 1 reproduced above), Petitioner provides evidence that each cluster node module sends commands to other cluster node modules such that they communicate with each other in a peer-to-peer fashion. *See* Pet. 32–35 (citing Ex. 1008, 17–18, Figs. 1 (arrows showing communications between dist.Scheduler), 5; Ex. 1010, 5, Fig. 5).

As one example, Petitioner explains as follows:

Schreiner1 teaches that the scheduler components of the cluster node modules implement peer-to-peer communication comprising direct message passing between peer nodes: “[A]ll nodes know of each other, i.e. a node knows the address of a machine and the number of a port on which (a thread of) the remote scheduler is listening for connection requests. When a node needs to send a message to one of its peers, it can thus establish a direct connection for message transfers.”

Pet. 33 (quoting Ex. 1008, 13; citing Ex. 1008, 17; Ex. 1005 ¶ 86).

As noted above, claim 1 also recites limitation c.iv, “a plurality of cluster node modules . . . configured . . . to accept instructions from the user interface or a script, and to interpret at least some of the user instructions such that the plurality of cluster node modules communicate with one another in order to act as a cluster.” In addition to the user interface identified in Schreiner1's Figure 1 (Ex. 1008, 9), Petitioner quotes Schreiner1 as disclosing a user interface. Pet. 37 (“The user interacts with Distributed Maple via a conventional Maple frontend (text or graphical), i.e. she operates within the familiar Maple environment for writing and executing parallel programs.” (quoting Ex. 1008, 7–8; citing *id.* at 3–4)).

Regarding the limitation of the cluster node modules “accept[ing] instructions” from the user interface, according to Petitioner, “[t]he root

IPR2021-00075
Patent 8,140,612 B2

kernel receives instructions from this user interface and passes them to its cluster node module.” *Id.* Petitioner explains as follows:

Fig. 1 . . . shows a command from “user interface” to the root kernel. If the command is a Distributed Maple instruction (e.g., dist[all], dist[start], dist[wait], etc.), the dist.maple component accepts the instruction and passes it to the scheduler component, as shown by the arrow directed from dist.maple to the scheduler component of the root cluster node module.

Pet. 37.

Relying further on Schreiner1, Petitioner provides the example of a “dist[all]” command sent from a user interface and interpreted by cluster node modules such that they act as a cluster:

The instructions from the user interface are interpreted by the cluster node modules “such that the plurality of cluster node modules communicate with one another in order to act as a cluster,” as the claim requires. As an example, Schreiner1 describes a user entering the Distributed Maple “dist[all]” instruction into the user interface: “dist[all](*command*) lets the Maple statement *command* be executed on every Maple kernel connected to the distributed session.” EX-1008, [9]; EX-1009, 30. A POSITA understands that, in order for this Distributed Maple instruction and its embedded Maple statement (“*command*”) to be executed on every kernel, the instruction is sent to, and accepted by, every cluster node module, which then interprets it in order to execute it. Once each node evaluates the Maple statement, it communicates the result back to the node that issued the dist[all] instruction. Accordingly, the cluster node modules interpret the dist[all] command such that the nodes communicate to evaluate a Maple statement in parallel, thereby satisfying this claim element. EX-1005, ¶93.

Pet. 38–39.

Petitioner provides another example as disclosed in Scheiner1. *See* Pet. 39–40 (relying on a “dist[start] instruction” entered from a “root node user interface,” and “interpreted by the root dist.maple component as a *task*,

IPR2021-00075
 Patent 8,140,612 B2

which the root scheduler can pass to other cluster nodes for execution.”
 (citing Ex. 1008, 8, 10–12, 14)).

Patent Owner does not dispute in a clear fashion that Schreiner1 teaches limitation [c.iv] under our preliminary claim construction set forth above (*supra* § II.C). See Prelim. Resp. 27 (“At best, the Petition suggests that the prior art processes instructions in the following order: user interface → kernel → cluster node module.”). Rather, Patent Owner relies on its claim construction argument, asserting that claim 1 precludes passing an instruction indirectly from a user interface, via a kernel, to a cluster node module. See Prelim. Resp. 27 (“This command processing order [in Schreiner1] is fundamentally different from the order required by claim 1: user interface → cluster node module → (kernels).”). As set forth above, the preliminary record does not support Patent Owner’s narrow claim construction that effectively requires a direct connection between the user interface and cluster node modules. See *supra* § II.C.

Patent Owner also argues that in Schreiner1, the root node is a master node and it schedules tasks between other nodes. See Prelim. Resp. 29–30. Therefore, according to Patent Owner, Schreiner1 does not disclose a “cluster node module,” because under Patent Owner’s claim construction, a cluster node module must “communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.” See *id.*; see also *supra* § II.C. However, as determined above for purposes of institution, the record does not support Patent Owner’s claim construction and does not require “communicating . . . without the tasks and data being required to go through a central server or master node.” See *supra* § II.C.

IPR2021-00075
 Patent 8,140,612 B2

Patent Owner also argues that Petitioner’s “cited pages never suggest that the user interface shown in Figure 1 of Schreiner 1 sends any instruction to the dist.maple component.” Prelim. Resp. 26 (citing Pet. 23; Ex. 2001 ¶ 60). Patent Owner argues that “the user interface communicates with the ‘Maple’ component—which the Petition calls the ‘kernel’—*not* the dist.maple component.” *Id.* at 27.

However, as indicated above, Schreiner’s Figure 1 shows the user interface connected to the Maple kernel, which connects to dist.maple, and then to dist.Scheduler. *See* Ex. 1008, 9 (Fig. 1). So all communications from the user interface to dist.Scheduler (cluster node module) pass via the Maple kernel and dist.maple to dist.Scheduler. *See id.* Although as noted above, Patent Owner argues that Schreiner1’s user interface does not send commands to dist.Scheduler, this appears to be another form of Patent Owner’s claim construction argument. *See* Prelim. Resp. 26. In other words, Patent Owner does not argue clearly that dist.Scheduler does not receive commands indirectly from the user interface via a kernel. Rather, as noted above, Patent Owner focuses on its claim construction argument, stating that “[a]t best, the Petition suggests that the prior art processes instructions in the following order: user interface → kernel → cluster node module.” *Id.* at 27. This is sufficient for purposes of institution under the claim construction set forth above. *See supra* § II.C.

To the extent Patent Owner argues that the dist.Scheduler does not even receive indirect instructions via a kernel from a user interface, Petitioner sufficiently shows otherwise for purposes of institution. As quoted above, Schreiner1 states that “[t]he user interacts with Distributed Maple via a conventional Maple frontend (text or graphical), i.e. she

IPR2021-00075
 Patent 8,140,612 B2

operates within the familiar Maple environment for writing and executing parallel programs.” Pet. 37 (quoting Ex. 1008, 7–8; citing *id.* at 3–4)). Petitioner quotes Schreiner² to show that the `dist.maple` component “run[s] on each Maple kernel” and “implements the interface between Maple and the scheduler.” *Id.* at 29 (quoting Ex. 1009, 24). The `dist.Scheduler` “manages the node interaction” and “[t]he initial scheduler process reads all application-specific information.” *Id.* (quoting Ex. 1009, 24). Further, “[c]ommunication between both components [i.e., the `dist.Scheduler` and `dist.maple`] is based on pipes (named pipes for the Maple kernel connected to the user interface and standard input/output streams for the backend kernels).” *Id.* (quoting Ex. 1009, 24). And “[t]he Maple kernel is a single-threaded process which communicates by a simple communication protocol with the scheduler on the same node. All capabilities for parallel and distributed program execution are embedded in this scheduler.” *Id.* at 28 (quoting Ex. 1008, 12).

In light of these cited or quoted teachings in the Petition, Petitioner sufficiently explains for purposes of institution that “the root cluster node module comprising the scheduler and `dist.maple` components accepts these Distributed Maple instructions directly from the user interface of its Maple kernel (or from a script written in the Maple language) and interprets those instructions.” Pet. 36–37 (citing Ex. 1008, 9–12; Ex. 1005 ¶ 90).

Further supporting the Petition, Schreiner¹ states “tasks created with `dist[start]` are scheduled on a fixed number of Maple kernels.” Ex. 1008, 10; *see* Pet. 36–37 (citing Ex. 1008, 9–12; Ex. 1005 ¶ 90). Also, “every scheduler instance accepts tasks from the attached computation kernel and schedules these tasks among all machines connected to the session.”

IPR2021-00075
Patent 8,140,612 B2

Ex. 1008, 9 (describing Schreiner1’s Fig. 1). Therefore, Schreiner1’s descriptions, including Figure 1, reveal sufficiently for purposes of institution that the cluster node module scheduler at least receives task or application instructions from the user interface, because it reads all application instructions, and it also distributes tasks to multiple Maple kernels based on a dist[start] command from a user interface. *See also* Pet. 28–29 (citing Ex. 1008, 17–18 (“§ 3.5.1: providing an example of message passing between kernels, dist.maple components, and dist.Scheduler components”); Ex. 1009, 24).

The final limitations of claim 1 follow:

- [d.] a communications network to connect the nodes;
- [e.] wherein the plurality of cluster node modules are configured to cooperate to interpret, translate, or interpret and translate commands for execution by a plurality of single-node kernel modules, and
- [f.] wherein at least one of the plurality of cluster node modules returns a result to the user interface.

For limitation d, Petitioner quotes Schreiner1’s disclosure of “networked environments” for the Distributed Maple software. *See* Pet. 42 (quoting Ex. 1008, 3). Petitioner also relies on other teachings in Schreiner1 describing computer nodes with processors connected in a network. *See id.* at 42–43 (citing Ex. 1008, 23, 25; Ex. 1005 ¶ 102).

For limitation e, Petitioner relies partly on its discussion of interpreting and translating Distributed Maple commands in connection with limitations b.ii, c.iii, and c.iv above. *See* Pet. 43–46. According to Petitioner, “the dist.maple and scheduler components interpret dist[all] and dist[start] commands for execution by the plurality of Maple kernels.” *Id.* at

IPR2021-00075
 Patent 8,140,612 B2

43. Petitioner also relies on the following load balancing disclosure for “cooperation” as claimed:

Moreover, “[a]ll remote schedulers send new tasks to the root node scheduler which distributes them among all machines.” EX-1008, [14]. The root scheduler (i.e., cluster node module) uses a load balancing scheme to distribute tasks. *Id.* Under this scheme, “a remote scheduler asks for new tasks whenever the number of received but not yet started tasks falls below a lower bound.” *Id.* This is yet another example of cooperation – in order for a task to be executed, the cluster node modules cooperate to establish a load balancing scheme to distribute and interpret or translate task commands for execution. EX-1005, ¶107.

Id. at 46.

For limitation f, Petitioner reproduces “a screenshot of a session in which the user inputs several instructions (highlighted in green) and the cluster node modules then return the result to the user interface (highlighted in orange).” *See* Pet. 47 (citing Ex. 1008, 8; Ex. 1009, 5; Ex. 1005 ¶ 110). Petitioner also relies on a graphical output example for display in Schreiner1’s user interface. *See id.* at 48 (reproducing Ex. 1008, Fig. 13).

b. Public Accessibility of the Distributed Maple Code

Patent Owner also generally argues, with respect to all of the claim limitations, that Petitioner failed to show that the Distributed Maple Code references were publically accessible prior to the date of the invention. *See* Prelim. Resp. 22–25 (noting that Distributed Maple Code includes Exhibits 1012–1018). However, Petitioner relies on the Distributed Maple Code references, which describe the source code for the Distributed Maple Code referenced in Schreiner1, Schreiner2, Schreiner3, and the Maple Guide, to *support* its showing based on the latter references. *See, e.g.*, Pet. 39 (“This understanding is further confirmed in the dist.maple source code files.”). As

IPR2021-00075
 Patent 8,140,612 B2

noted above, Petitioner also argues “Schreiner1 expressly teaches nearly all of the claim limitations by itself.” *Id.* at 16 (citing Ex. 1005 ¶ 48).

Patent Owner does not directly argue that the other references do not support institution sufficiently without the Distributed Maple Code (i.e., source code). In any event, addressing the Distributed Maple Code, Patent Owner contends that Petitioner relies on the uncorroborated testimony of Dr. Schreiner that he posted the Distributed Maple Code “in 2003 on the public website of Research Institute for Symbolic Computation (‘RISC’), where the references allegedly could be downloaded through the webpage shown in Exhibit 1024.” Prelim. Resp. 22–23 (arguing “corroboration is required of a witness’s testimony about his own allegedly invalidating activities” (citing *Finnigan Corp. v. ITC*, 180 F.3d 1354, 1366 (Fed. Cir. 1999))). However, Petitioner also cites the website as published in Schreiner1, as Patent Owner acknowledges. Prelim. Resp. 23; Ex. 1008, 6. Nevertheless, Patent Owner contends that “[t]he most the evidence submitted by Petitioner shows is that [Dr.] Schreiner himself, or possibly others who helped create the Distributed Maple Code or already knew of its existence, may have been able to locate whatever version was posted at that time.” *Id.* at 24.

This line of argument downplays that Schreiner1, published in the Journal of Symbolic Computation in 2003, would have pointed interested artisans to the website listed therein in order to obtain the “Distributed Maple system itself,” the main subject of Schreiner1 and described as “freely available.” *See* Ex. 1008, 5. Also, the Internet Archive screenshot, Exhibit 1024, describes “Distributed Maple” and lists the same website as published in Schreiner1, and describes the website as “[m]aintained by: Wolfgang Schreiner, Last Modification: July 14, 2003.” *See also* Ex. 1006

IPR2021-00075
Patent 8,140,612 B2

¶ 24 (Dr. Schreiner noting that the Internet Archive screenshot states “Last Modification: July 14, 2003” and testifying “[t]hat is consistent with my recollection of the time when I last modified this page” (citing Ex. 1024)); Ex. 1025 (similar Internet Archive screenshot evidence). This evidence corroborates Dr. Schreiner’s testimony as to the timeframe he uploaded the source code to the RISC website.

Dr. Schreiner also testifies that “[i]t has been my practice to check, from time to time, whether my software was accessible through Google search results, and I did this prior to 2005 for these particular web pages and confirm that my Distributed Maple papers and software were accessible through Google searches.” Ex. 1006 ¶ 21. Patent Owner argues that “Petitioner relies entirely upon Schreiner’s memory from more than *seven* years ago to suggest the version of Distributed Maple Code filed in this IPR was publicly accessible back then.”⁶ Prelim. Resp. 23 (emphasis added). However, as indicated above, Schreiner1, coauthored by Dr. Schreiner with others, and the Internet Archive documents, corroborate Dr. Schreiner’s testimony about uploading the software on the RISC website.⁷ During trial, Patent Owner will have the opportunity to cross-examine Dr. Schreiner, including regarding Google searches and his memory, assuming for the sake of argument that the ability to search the RISC website using Google is relevant to show public accessibility of the source code.

⁶ Given the timeframe of 2003 to 2005 at issue here, Patent Owner probably meant “seventeen years ago” instead of “seven years ago.”

⁷ As noted above (§ II.D.1), two others co-authored Schreiner1 with Dr. Schreiner.

IPR2021-00075
 Patent 8,140,612 B2

On this preliminary record, sufficient evidence exists to show that the Distributed Maple Code was publically available in 2003 and thereafter up to the date of the invention in 2006. *See* Ex. 1001, code (60) (listing the filing date of a provisional application as October 11, 2006). Patent Owner does not dispute that some form of the source code existed prior to the date of the invention. To the extent the source code may have changed over the relevant timeframe as Patent Owner argues (*see* Prelim. Resp. 23), the parties will have the opportunity to address the materiality of any such changes as they relate to specific claim limitations or evidence during trial.

Even if the source code was not publically available at the relevant time, Petitioner’s reliance on it as extrinsic evidence solely to support its showing of how the Distributed Maple system operated at the time of the invention would be proper. *See In re Baxter Travenol Labs.*, 952 F.2d 388, 390 (Fed. Cir. 1991) (extrinsic evidence may be used to explain what a reference discloses); *Hospira v. Fresenius Kabi USA*, 946 F.3d 1322, 1329 (Fed. Cir. 2020) (“[E]xtrinsic evidence can be used to demonstrate what is ‘necessarily present’ in a prior art embodiment even if the extrinsic evidence is not itself prior art.”). In any event, on this preliminary record for purposes of institution, after considering Petitioner’s showing and Patent Owner’s arguments and objective evidence of nonobviousness (as discussed further below), Petitioner sufficiently shows that Schreiner1, Schreiner2, Schreiner3, and the Maple Guide teach claim 1 with or without the supporting source code as disclosed in the Distributed Maple Code.⁸

⁸ Patent Owner also contends that Petitioner’s declarant impermissibly relies on “public use” information—i.e., information about commercial embodiments. *See* Prelim. Resp. 37–38; PO Sur-reply 10. Petitioner disagrees. Pet. Reply 9–10. For purposes of institution on this preliminary

IPR2021-00075
 Patent 8,140,612 B2

c. Alleged Objective Evidence of Nonobviousness

Objective evidence of nonobviousness “may often be the most probative and cogent evidence in the record” and “may often establish that an invention appearing to have been obvious in light of the prior art was not.” *Transocean Offshore Deepwater Drilling, Inc. v. Maersk Drilling USA, Inc.*, 699 F.3d 1340, 1349 (Fed. Cir. 2012) (citing *Stratoflex, Inc. v. Aeroquip Corp.*, 713 F.2d 1530, 1538 (Fed. Cir. 1983)).

Patent Owner argues that objective evidence regarding its SEM and SET products supports the nonobviousness of the challenged claims. Prelim. Resp. 33–50. Patent Owner alleges evidence of long-felt and unresolved need, failure by others, praise by others, skepticism of others, and copying. *Id.*

(i.) Nexus

Nexus is a legally and factually sufficient connection between the objective evidence and the claimed invention requiring a tribunal to consider the objective evidence in determining nonobviousness. *Demaco Corp. v. F. Von Langsdorff Licensing Ltd.*, 851 F.2d 1387, 1392 (Fed. Cir. 1988) (“Once a prima facie case of nexus is made the court must consider the evidence adduced on both sides of the question, with such weight as is

record, Petitioner sufficiently shows that the cited Exhibits support Dr. Tufo’s testimony and at least teach claim 1 without improper reliance on public use information. *See id.* (mapping support for Dr. Tufo’s testimony at Ex. 1005 ¶¶ 37–40, 70–76 to Ex. 1008–Ex. 1011). (To provide further support for its position, Petitioner also cites an alleged Exhibit supplied by Patent Owner (i.e., “EX-2005, 2”). *See* Pet. Reply 10. However, this Exhibit does not appear in the record so any argument premised on it provides no added support for Petitioner.)

IPR2021-00075
Patent 8,140,612 B2

warranted.”). “The patentee bears the burden of showing that a nexus exists” *WMS Gaming Inc. v. Int’l Game Tech.*, 184 F.3d 1339, 1359 (Fed. Cir. 1999).

Patent Owner contends that “traditional parallel-computing architectures was notoriously difficult, time-consuming, and expensive.” Prelim. Resp. 34 (citing Ex. 2033 ¶¶ 13–18; Ex. 2007 ¶ 8; Ex. 2008 ¶ 21). Patent Owner contends that programmers typically generated parallel code by breaking up and converting serial code “for execution on each of the nodes of a parallel computer.” *Id.* According to Patent Owner, “there was a long-felt but unmet need for a way to unlock the performance advantages of cluster computing without requiring specialized expertise or excessive time, effort, and cost (i.e., a need to break the performance/programming barrier).” *Id.* at 35 (citing Ex. 2033 ¶¶ 10–19; Ex. 2007 ¶¶ 8, 11; Ex. 2008 ¶¶ 16–23).

Patent Owner alleges it developed “a cluster-computing architecture, used in its SEMTM and SETTM’s products, that met this long-felt, unmet need.” Prelim. Resp. 35–36. Patent Owner asserts that SEMTM and SETTM “enable[] parallel execution of Mathematica and more general applications, respectively . . . without extensive specialized programming expertise, or the excessive investment of time and effort demanded by traditional parallel computing architectures.” *Id.* at 36–37 (citing Ex. 2033 ¶¶ 20–30; Ex. 2007 ¶¶ 10, 12 (discussing SEMTM); Ex. 2008 ¶¶ 25–28 (discussing SETTM)).

As further described by Patent Owner, to meet this “long-felt, unmet need,” “[t]he new architecture interposed a communication layer between the front end user interface and the kernels running on each node of the cluster, or back end.” Prelim. Resp. 35–36. Patent Owner contends that

IPR2021-00075
 Patent 8,140,612 B2

“[t]he unique SEMTM and SETTM cluster-computing architecture embodied by the challenged claims is the reason that SEMTM and SETTM were able to meet the long-felt but previously unmet need.” *Id.* at 37 (citing Ex. 2033 ¶¶ 24, 27–29; Ex. 2001 ¶¶ 83–88). Patent Owner argues this alleged claimed architecture obtained “superior performance” and “ease of use” that “were unexpected” and “surprising.” *Id.* at 38. Similarly, Patent Owner argues that “SEMTM and SETTM succeeded where others failed” and received praise, and “many were skeptical that the SEMTM architecture could truly enable the high performance of cluster computing without requiring specialized expertise or excessive time and effort.” *Id.* at 38–40.

Patent Owner argues that “[t]here is sufficient nexus between the objective evidence related to SEMTM and SETTM and the challenged claims” because the “SEMTM and SETTM products practice at least the challenged independent claims.” Prelim. Resp. 41. Patent Owner argues that “[e]ach of the objective indicia of non-obviousness results from the SEMTM and SETTM architecture embodied by the challenged claims.” *Id.* at 45. Patent Owner argues that “[o]ther parallel-computing architectures failed to meet the long-felt, unmet need precisely because they lacked SEMTM and SETTM’s claimed architecture.” *Id.* at 45 (citing Ex. 2033 ¶ 37). Patent Owner argues the “claimed architecture” is responsible for all of its asserted objective indicia of nonobviousness. *Id.* (citing Ex. 2033 ¶¶ 39–42, 46–47).

Nevertheless, even if the SEMTM and SETTM products fall within the broad scope of claim 1, for similar reasons to those underlying the claim construction as set forth above (§ II.C), the challenged claims do not require the “unique . . . architecture” of SEMTM and SETTM. *See* Prelim. Resp. 37. In simple terms, the challenged claims are not architecture-specific. *See*

IPR2021-00075
Patent 8,140,612 B2

MeadWestVaco Corp. v. Rexam Beauty and Closures, Inc., 731 F.3d 1258, 1264 (Fed. Cir. 2013) (error to consider “*secondary considerations of non-obvious [that] involved only fragrance-specific uses*,” when “*claims now at issue are not fragrance-specific*” (emphasis added)). The court in *MeadWestVaco* held that the district court erred because it “credited evidence advanced to show long-felt need and commercial success specific to the perfume industry,” and the claims were not limited to fragrance-specific dispensers. *See id.* (reasoning that ““objective evidence of non-obviousness must be commensurate in scope with the claims which the evidence is offered to support””) (quoting *Asyst Techs., Inc. v. Emtrak, Inc.*, 544 F.3d 1310, 1316 (Fed. Cir. 2008) (internal quote citation omitted)); *see also Brown & Williamson Tobacco Corp. v. Philip Morris Inc.*, 229 F.3d 1120, 1130 (Fed. Cir. 2000) (stating the presumption that commercial success is due to the patented invention applies “if the marketed product embodies the claimed features, and is coextensive with them”).

The Federal Circuit recently addressed nexus in two related Board cases, *Fox Factory, Inc. v. SRAM, LLC*, 944 F.3d 1366, 1373–78 (Fed. Cir. 2019) (“*Fox Factory I*”) and *Fox Factory, Inc. v. SRAM, LLC*, 813 F. App’x 539, 542 (Fed. Cir. 2020) (*Fox Factory II*). In *Fox Factory II*, the court characterized the Board’s holding underlying *Fox Factory I*, as follows:

In [*Fox Factory I*], this court held that the Board misapplied the legal requirement, incumbent upon patent owners, of showing a nexus between evidence of secondary considerations and the obviousness of the claims of that patent—in particular, the requirement that the product from which the secondary considerations arose is “coextensive” with the claimed invention. *Fox Factory*, 944 F.3d at 1373–78. Contrary to the Board’s view, we reaffirmed in that case that a product is not

IPR2021-00075
 Patent 8,140,612 B2

coextensive with a claimed invention simply because it falls within the scope of the claim.

Fox Factory II, 813 F. App'x at 542. In other words, no presumption of nexus exists for products that simply fall in the broad scope of the claims if the products are not coextensive with the claims.

As noted above, according to Patent Owner, “the new architecture interposed a communication layer between the front end user interface and the kernels running on each node of the cluster, or back end”—i.e., a “unique . . . architecture.” *See* Prelim. Resp. 35–37. As construed above, however, the claims do not require a communication layer between the user interface and kernels—the central feature relied upon by Patent Owner for its objective evidence of nonobviousness. *See supra* § II.C; Pet. Reply 6 (arguing that Dr. Dauger’s testimony “rests on an incorrect claim construction”). Therefore, on this preliminary record, because the claims do not require this allegedly “unique . . . architecture,” the claims are not coextensive (or reasonably commensurate in scope) with the products and other asserted evidence, so no nexus exists.

For the foregoing reasons and on this preliminary record, Patent Owner fails to meet its burden of establishing a nexus between the objective evidence regarding its SEM and SET products and the claims of the '612 patent. We, therefore, do not accord substantial weight to such evidence. *Fox Factory I*, 944 F.3d at 1373. For the sake of completeness, however, we address Patent Owner’s remaining allegations relating to objective indicia of nonobviousness.

IPR2021-00075
 Patent 8,140,612 B2

(ii) *Long-Felt Need and Failure of Others*

“The existence of a long-felt but unsolved need that is met by the claimed invention is . . . objective evidence of non-obviousness.”

Millennium Pharms., Inc. v. Sandoz Inc., 862 F.3d 1356, 1369 (Fed. Cir. 2017) (citing *In re Cyclobenzaprine Hydrochloride Extended-Release Capsule Patent Litig.*, 676 F.3d 1063, 1081–83 (Fed. Cir. 2012)).

“Long[-]felt need is closely related to the failure of others. Evidence is particularly probative of obviousness when it demonstrates both that a demand existed for the patented invention, and that others tried but failed to satisfy that demand.” *Cyclobenzaprine*, 676 F.3d at 1082.

Patent Owner argues “there was a long-felt but unmet need for a way to unlock the performance advantages of cluster computing without requiring specialized expertise or excessive time, effort, and cost.” Prelim. Resp. 35. Patent Owner argues that its SEM and SET products met this need. *Id.* at 35–37 (citing Ex. 2001 ¶¶ 83–88; Ex. 2007 ¶¶ 10, 12; Ex. 2008 ¶¶ 25–28; Ex. 2033 ¶¶ 10–30).

Petitioner argues that “[Dr.] Dauger provides no evidence or explanation for why interposing the communications software between the user interface and the kernel, as compared to connecting the communications software in some other way, would make any difference at all to the user.” Pet. Reply 6. Petitioner argues that this testimony is also unpersuasive because it relies on an incorrect claim interpretation that requires the cluster node modules to accept instructions from the user interface without the instructions first passing through any kernel. *Id.* at 6–8.

Patent Owner argues that Petitioner’s arguments regarding claim construction are outside the scope of our Order authorizing Petitioner to file

IPR2021-00075
Patent 8,140,612 B2

its Reply. PO Sur-reply 6–9. To the contrary, as discussed in considering nexus above (§ II.D.6.c.ii), the scope of the claims is relevant to objective indicia of nonobviousness.

Regarding the failure of others, Patent Owner acknowledges that others developed automatic parallelizers and universal compilers that converted serial code to parallel code, but argues that none achieved performance comparable to traditional parallel-computing architectures. Prelim. Resp. 39; *see also* PO Sur-reply 3.

Petitioner argues that “P[atent] O[wner]’s argument is irrelevant because the claims neither recite ‘automatic’ or ‘universal’ parallelization nor require a specific level of optimization or advantageousness.” Pet. Reply 2 (citing *ABT Sys., LLC v. Emerson Elec. Co.*, 797 F.3d 1350, 1362 (Fed. Cir. 2015)).

Patent Owner relies on the testimony of Dr. Dager, Dr. Bhansali, and Mr. Bancroft to support its arguments about a long-felt, unmet need. *See* Prelim. Resp. 33–37. However, none of these declarants establishes that others unsuccessfully attempted to solve the problem. Dr. Dager states that “[n]umerous others had tried to implement automatic parallelizers or universal compilers that would take serial object code as input and output object parallel code,” but “[n]one of these efforts succeeded to produce accurate parallel code that was sufficiently optimized or advantageous enough to catch on.” Ex. 2033 ¶ 37. These conclusory and uncorroborated statements fail to establish that others actually tried to solve the asserted problem. The testimony of the other declarants fares no better. Dr. Bhansali merely states that he was not aware of any other products like the SEM product. Ex. 2007 ¶ 11. Mr. Bancroft states that he “had heard rumo[rrs] of

IPR2021-00075
 Patent 8,140,612 B2

people trying to develop a universal parallelizer that could be used to automatically parallelize serial code.” Ex. 2008 ¶ 30. None of this testimony persuasively establishes that others actually tried and failed to solve the problem asserted by Patent Owner.

Accordingly, for the foregoing reasons and on this preliminary record, Patent Owner’s evidence of long-felt need and failure of others is weak.

(iii) Unexpected Results

“If a patent challenger makes a prima facie showing of obviousness, the owner may rebut based on ‘unexpected results’ by demonstrating ‘that the claimed invention exhibits some superior property or advantage that a person of ordinary skill in the relevant art would have found surprising or unexpected.’” *Procter & Gamble Co. v. Teva Pharms. USA, Inc.*, 566 F.3d 989, 994 (Fed. Cir. 2009) (quoting *In re Soni*, 54 F.3d 746, 750 (Fed. Cir. 1995)). “To be particularly probative, evidence of unexpected results must establish that there is a difference between the results obtained and those of the closest prior art, and that the difference would not have been expected by one of ordinary skill in the art at the time of the invention.” *Bristol-Myers Squibb Co. v. Teva Pharms. USA, Inc.*, 752 F.3d 967, 977 (Fed. Cir. 2014); *see also Kao Corp. v. Unilever U.S., Inc.*, 441 F.3d 963, 970 (Fed. Cir. 2006) (“[W]hen unexpected results are used as evidence of nonobviousness, the results must be shown to be unexpected compared with the closest prior art.”) (quoting *In re Baxter Travenol Labs.*, 952 F.2d at 392).

Patent Owner argues that the performance and ease of use of its SEM and SET products was unexpected. Prelim. Resp. 37–39 (citing Ex. 2033 ¶¶ 38–41). Patent Owner argues that its SEM product outperformed gridMathematica. *Id.* at 37–38. Patent Owner argues that it was able to

IPR2021-00075
Patent 8,140,612 B2

parallelize Wolfram Research's Mathematica, Apple's HD QuickTime Exporter, and Equalis's Scilab using its SET product in a much shorter time period than expected. *Id.* at 38.

Petitioner argues that Patent Owner's reliance on the testimony of Dr. Dauger is unpersuasive because Dr. Dauger is a listed inventor of the '612 patent. Pet. Reply 1. Petitioner also argues that Dr. Dauger's "testimony is also irrelevant because it compares the claimed invention against gridMathematica, not the 'closest prior art.'" *Id.* (citing *In re Harris*, 409 F.3d 1339, 1344 (Fed. Cir. 2005); *Trs. of Columbia Univ. v. Illumina, Inc.*, 620 F. App'x 916, 922 (Fed. Cir. 2015)).

Patent Owner replies that "Dr. Dauger's testimony was submitted under oath and penalty of perjury" and the other evidence cited in its Preliminary Response support its contention that the SEM and SET products exhibited surprising results. PO Sur-reply 1–2. Patent Owner also argues that it "chose the closest prior art by comparing SEM™ with gridMathematica and SET™ with the conventional method of parallelizing applications." *Id.* at 2 (citing Ex. 2033 ¶ 40).

Patent Owner relies almost exclusively on the testimony of Dr. Dauger in asserting the surprising results of the SEM and SET products. *See* Prelim. Resp. 37–39 (citing Ex. 2033 ¶¶ 38–41). Dr. Dauger testifies the he was surprised that the SEM product performed better than gridMathematica and that Patent Owner was able to parallelize Mathematica "in one man-month." Ex. 2033 ¶¶ 38–41. Dr. Dauger is an inventor of the '612 patent, was Patent Owner's CTO, and is currently a consultant employed by Patent Owner. Ex. 1001, code (75); Ex. 2033 ¶ 1; Ex. 2015, 2; Ex. 2018, 2. On this preliminary record, Dr. Dauger's testimony about his

IPR2021-00075
Patent 8,140,612 B2

personal surprise at the SEM and SET products that he helped create unpersuasive to establish unexpected results of these products. *See In re Cree*, 818 F.3d 694, 702 (Fed. Cir. 2016) (citing *Power-One v. Artesyn Techs., Inc.*, 599 F.3d 1343, 1352 (Fed. Cir. 2010) (concluding that “self-serving statements from researchers about their own work” do not have the same credibility as statements made by disinterested parties). Notably, Patent Owner provides no evidence to corroborate Dr. Dauger’s assertions of unexpected results.

Additionally, Patent Owner provides no comparative testing against any prior art configuration, be it the closest or otherwise. Although Dr. Dauger testifies that some testing was performed (*see* Ex. 2033 ¶ 39), no documentation or data are provided from that testing to substantiate his assertions. This is the type of conclusory evidence that has been found insufficient. *See, e.g., In re Lindner*, 457 F.2d 506, 508 (CCPA 1972) (“This court has said previously that mere lawyers’ arguments unsupported by factual evidence are insufficient to establish unexpected results. . . . Likewise, mere conclusory statements in the specification and affidavits are entitled to little weight when the Patent Office questions the efficacy of those statements.”).

Furthermore, Patent Owner does not persuasively argue that gridMathematica is the closest prior art. As noted by Petitioner, Patent Owner does not compare its products to the asserted references or the Distributed Maple system disclosed therein.

Accordingly, for the foregoing reasons and on this preliminary record, Patent Owner’s evidence of unexpected results is weak.

IPR2021-00075
 Patent 8,140,612 B2

(iv) Industry Praise

“Evidence that the industry praised a claimed invention or a product that embodies the patent claims weighs against an assertion that the same claimed invention would have been obvious. Industry participants, especially competitors, are not likely to praise an obvious advance over the known art.” *Apple Inc. v. Samsung Elecs. Co.*, 839 F.3d 1034, 1053 (Fed. Cir. 2016) (en banc).

Relying on the asserted statements of Dr. Bhansali and Yuko Matsuda, Patent Owner argues that industry praise supports patentability of the ’612 patent claims. Prelim. Resp. 39–40. According to Patent Owner, Dr. Bhansali found the SEM product to be efficient for load balancing issues and Mr. Matsuda endorsed the SEM product. *Id.*

Petitioner argues that “[Dr.] Bhansali focuses on ‘load balancing,’” which “has no nexus because the patents do not assert that load balancing was novel or non-obvious.” Pet. Reply 3–4 (citing *Kennametal, Inc. v. Ingersoll Cutting Tool Co.*, 780 F.3d 1376, 1385 (Fed. Cir. 2015)). Petitioner argues that Patent Owner’s proposed construction of “cluster node module” excludes the only method of load balancing disclosed in the ’612 patent. *Id.* at 4–5. Petitioner also argues that the references asserted in the Petition teach load balancing. *Id.* at 4.

Patent Owner argues that Petitioner’s arguments regarding claim construction are outside the scope of our Order authorizing Petitioner to file its Reply. PO Sur-reply 1, 4–6. To the contrary, as discussed in considering nexus above (§ II.D.6.c.ii), the scope of the claims is relevant to objective indicia of nonobviousness.

IPR2021-00075
 Patent 8,140,612 B2

In order for evidence of industry praise to be probative of nonobviousness, the evidence must relate specifically to features of the claimed invention. *See Apple*, 839 F.3d at 1053–55 (discussing “substantial evidence of praise in the industry that specifically related to features of the claimed invention”). As argued by Patent Owner, Dr. Bhansali testifies that he found the SEM product to be efficient for load balancing issues, by which he means “*the distribution of different parts of an algorithm or application across different nodes* and the overall process of parallelizing the algorithm or application.” Ex. 2007 ¶ 12 (emphasis added). The ’612 patent refers to load balancing in a similar manner. *See* Ex. 1001, 21:8–55. As correctly noted by Petitioner, the challenged claims do not recite load balancing or otherwise require distributing commands among the nodes in a particular manner. Therefore, on this preliminary record, Dr. Bhansali’s testimony appears to focus on unclaimed features such that it is not probative of nonobviousness.

Regarding the asserted statements made by Mr. Matsuda, Patent Owner cites to the Dauger Declaration rather than any submission endorsed by Mr. Matsuda. Prelim. Resp. 40 (citing Ex. 2033 ¶¶ 44–45). Dr. Dauber cites to a slide deck which he appears to have prepared and the substance of which consists only of two quotations. Ex. 2033 ¶ 44 (citing Ex. 2018, 4); *see also* Ex. 2018, 2 (listing Dean E. Dauger, Ph.D. as the author). On this record, the uncorroborated statements of Dr. Dauber are unpersuasive to support the asserted statement of Mr. Matsuda.

Dr. Dauger also cites to Exhibit 2023, referring to it as a “white paper” written by Mr. Matsuda. Ex. 2033 ¶ 45 (citing Ex. 2023, 2). Initially, it is not clear what significance a “white paper” carries. In any

IPR2021-00075
 Patent 8,140,612 B2

event, in the sentence cited by Patent Owner, Mr. Matsuda merely states that the SEM product “stands in an advantageous position” compared with an undefined “Parallel Computing Toolkit” when used with Mathematica. Ex. 2023, 2. This is not the type of competitor praise that courts have found to be indicative of non-obviousness. *See, e.g., Apple*, 839 F.3d at 1053–54 (discussing “numerous internal Samsung documents that both praised Apple’s slide to unlock feature and indicated that Samsung should modify its own phones to incorporate Apple’s slide to unlock feature”).

Accordingly, for the foregoing reasons and on this preliminary record, Patent Owner’s evidence of industry praise is weak.

(v) Skepticism

Evidence of industry skepticism weighs in favor of nonobviousness. *See United States v. Adams*, 383 U.S. 39, 52 (1966). “If industry participants or skilled artisans are skeptical about whether or how a problem could be solved or the workability of the claimed solution, it favors nonobviousness.” *WBIP, LLC v. Kohler Co.*, 829 F.3d 1317, 1335 (Fed. Cir. 2016).

Patent Owner argues that experts expressed skepticism that the SEM and SET products would work. Prelim. Resp. 40–41. Regarding the SEM product, Patent Owner relies solely on the testimony of Dr. Bhansali and Mr. Bancroft. *Id.* at 41 (citing Ex. 2007 ¶ 8; Ex. 2008 ¶ 32). Regarding the SET product, Patent Owner relies on the opinion of one unidentified Department of Energy (“DOE”) “reviewer.” *Id.* at 40 (citing Ex. 2033 ¶ 52); *see also* PO Sur-reply 3–4.

Petitioner argues that rather than expressing skepticism that Patent Owner’s products would work, the DOE reviewers quoted in Patent Owner’s

IPR2021-00075
 Patent 8,140,612 B2

exhibits “expressed skepticism over the bold performance claims made by P[atent] O[wner].” Pet. Reply 2–3.

Regarding the SEM product, Patent Owner relies solely on statements of Dr. Bhansali and Mr. Bancroft regarding their personal experience with the SEM product. Prelim. Resp. 40 (citing Ex. 2007 ¶ 8; Ex. 2008 ¶ 32). Dr. Bhansali states that “[w]hen I first learned about SEM, I was uncertain whether the product would perform as promised.” Ex. 2007 ¶ 8. Dr. Bhansali states that he “graduated from Cal. Tech. with a dual B.S.-M.S. in physics, and engineering and applied science” and “received [a] Ph.D. in theoretical physics from Harvard University.” *Id.* ¶¶ 3–4. Neither Patent Owner nor Dr. Bhansali provide any detail about his industrial experience. Notably, Patent Owner did not file a CV for Dr. Bhansali in the record. Therefore, Patent Owner fails to establish that Dr. Bhansali is an artisan of ordinary skill with respect to parallel or distributed computing, and his opinion testimony carries little weight.

Although Mr. Bancroft states that he provided the SEM product to “many experienced parallel programmers” (Ex. 2008 ¶ 32), he provides no information about these allegedly experienced programmers. Additionally, neither Patent Owner nor Mr. Bancroft provide his CV, so it is difficult to assess his credibility to provide technical testimony. *See* Ex. 2008. Mr. Bancroft’s experience appears to involve business development matters rather than technical engineering or computer science research and development. *See id.* ¶¶ 4–14. Moreover, Mr. Bancroft is on Patent Owner’s Business Advisory Board. *Id.* ¶ 33. Therefore, it appears that Mr. Bancroft is not a disinterested party and may have economic or other

IPR2021-00075
 Patent 8,140,612 B2

interest in Patent Owner's success in this proceeding. Accordingly, on this preliminary record, Mr. Bancroft's opinion testimony carries little weight.

Regarding the SET product, Petitioner sufficiently shows that the DOE reviewers appear to indicate that the submissions they reviewed lacked sufficient detail for them to evaluate the performance assertions made in the submissions. *See* Pet. Reply 2–3. Patent Owner relies on the comments of “Reviewer 2” of Exhibit 2019. Prelim. Resp. 40 (citing Ex. 2033 ¶ 52; Ex. 2019, at 2). This reviewer states that “[t]he proposal . . . provides no quantitative or qualitative evidence of (efficient or not) use of compute[r] resou[r]ces by SET.” Ex. 2019, 2. This reviewer also states that “[t]he applicants have not demonstrated quantitatively that their technology provides real results. . . . SET may prove to be the great success the applicants suggest, but there is no proof that it works on real CAD/CAM/CAE applications.” *Id.* at 3. Continuing, this reviewer states that “[t]he applicants haven't clearly defined the nature of the plasma code, nor the effort in porting it to SET.” *Id.* Thus, the statements of Reviewer 2 appear to stem from a lack of detail to assess the credibility of the assertions in the submissions.⁹

Other reviewers similarly identify a lack of detail in the submissions. Ex. 2019, 4 (“[T]he main concepts of this proposal have not been presented in any substantial detail.” “There is no sound plan to showing that this SET-based approach can be commercially viable.”); Ex. 2021, 1 (“[T]here is no plan to compare the performance of the applications compared to their theoretical performance.”), 2 (“The auto parallelization tools have not

⁹ The submissions to the DOE are not of record in this case.

IPR2021-00075
 Patent 8,140,612 B2

provided high performance as they usually have too many generalizations to take advantage of a particular computing architecture. I do not have evidence that the SET tool is any different.”), 4 (“The applicant has provided a general outline of the comparison test approach, but further details in the work plan are needed.”; “While there is an overall projection of technical relevance, the lack of specificity in the proposed test situation presents a high level of uncertainty in achieving the more ambitious goals of this proposal.”); Ex. 2022, 3 (“The performance of SET in Linux and Mac OS operating environments, the type of efficiency increases achieved, and the strength and limitations of the SET approach are not adequately described.”), 4 (“The applicant has provided a general description of the technical problem and work plan, but specific details of the technical challenges to be encountered with the SET technology should be described in greater detail.”). Thus, to the extent one reviewer expressed skepticism that the SET product would perform as claimed, this evidence is undercut by the overwhelming expression of a lack of detail provided in the submitted proposals that were reviewed.

Accordingly, for the foregoing reasons and on this preliminary record, Patent Owner’s evidence of skepticism of others is weak.

(vi) Copying

“Copying may indeed be another form of flattering praise for inventive features.” *Crocs, Inc. v. ITC*, 598 F.3d 1294, 1311 (Fed. Cir. 2010). Copying “requires evidence of efforts to replicate a specific product.” *Wyers v. Master Lock Co.*, 616 F.3d 1231, 1246 (Fed. Cir. 2010). “This may be demonstrated either through internal documents; direct evidence such as disassembling a patented prototype, photographing its

IPR2021-00075
 Patent 8,140,612 B2

features, and using the photograph as a blueprint to build a virtually identical replica; or access to, and substantial similarity to, the patented product (as opposed to the patent).” *Iron Grip Barbell Co. v. USA Sports, Inc.*, 392 F.3d 1317, 1325 (Fed. Cir. 2004) (internal citations omitted). “We note, however, that a showing of copying is only equivocal evidence of nonobviousness in the absence of more compelling objective indicia of other secondary considerations.” *Ecolochem, Inc. v. S. Cal. Edison Co.*, 227 F.3d 1361, 1380 (Fed. Cir. 2000); *see also In re GPAC*, 57 F.3d 1573, 1580 (Fed. Cir. 1995) (“[M]ore than the mere fact of copying by an accused infringer is needed to make that action significant to a determination of the obviousness issue.” (quoting *Cable Elec. Prods. v. Genmark, Inc.*, 770 F.2d 1015, 1028 (Fed. Cir. 1985))); *Institut Pasteur & Université Pierre et Marie Curie v. Focarino*, 738 F.3d 1337, 1347–48 (Fed. Cir. 2013) (“Copying requires duplication of features of the patentee’s work based on access to that work, lest all infringement be mistakenly treated as copying. . . . But the Board did not analyze whether Pasteur’s showing of the similarities of its method to the content of the cited publications, *e.g.*, their use of the same specific GIIE endonuclease, indicated that the publications’ authors had access to, and borrowed from, the Pasteur sources.”); *Liqwd, Inc. v. L’Oreal USA, Inc.*, 941 F.3d 1133, 1136–39 (Fed. Cir. 2019) (mere access to information coupled with allegations of infringement are not sufficient evidence of copying, absent evidence of other circumstances, such as copying specifics of a disclosed or actual product, or altering a design after obtaining access to the information). “Of course, the proponent of objective evidence offered to show nonobviousness, such as copying, must show that a nexus exists

IPR2021-00075
Patent 8,140,612 B2

between the evidence *and the claimed features of the invention.*” *Liqwd*, 941 F.3d at 1138 (emphasis added).

Patent Owner asserts that it provided information to Petitioner regarding its SET product during a November 2012 meeting and in a subsequent “email attaching a specially tailored data sheet.” Prelim. Resp. 45–47 (citing Ex. 2033 ¶ 56). Patent Owner argues that “Petitioner then copied the claimed invention by incorporating the claimed architecture into Petitioner’s GPGPUs in the manner described by the datasheet” and “named its GPU interconnect architecture, which uses the claimed structure, NVLink™.”¹⁰ *Id.* at 46 (citing Ex. 2033 ¶ 59); *see also* PO Sur-reply 9–10.

Petitioner traverses Patent Owner’s assertions of copying, calling it “a gross misrepresentation to the PTAB.” Pet. Reply 8–9. Petitioner asserts that the inventors of the ’612 patent sent an unsolicited email to one of its employees attaching a public SET datasheet that “did not have any suggestion of ‘provid[ing] a communications infrastructure for direct all-to-all communications between each GPU.’” *Id.* at 8 (alteration in original) (citing Prelim. Resp. 56).

Patent Owner provides no evidence to support its assertion. Patent Owner does not provide any description of the NVLink product or compare this product to the claims of the ’612 patent. On this record, Patent Owner’s conclusory assertions are inadequate to establish copying of the claimed invention by Petitioner.

¹⁰ Dr. Dauger refers to NVIDIA’s “general-purpose GPU (‘GPGPU’) supercomputing” and describes “NVIDIA’S Tesla GPGPUs as hardware black boxes on which the back end executes.” Ex. 2033 ¶¶ 56–57 (citing Ex. 2020, 4, Fig. 3).

IPR2021-00075
 Patent 8,140,612 B2

Accordingly, for the foregoing reasons and on this preliminary record, Patent Owner's evidence of copying is weak.

d. Claims 6, 10, 12, 13, 15, and 29

Building on its showing with respect to claim 1, Petitioner presents a sufficient showing supported by the record with respect to dependent claims 6, 10, 12, 13, 15, and 29. *See* Pet. 48–59. Patent Owner does not address these dependent claims individually.

e. Summary

Based on the foregoing discussion, Petitioner sufficiently establishes for purposes of institution that the combination of Schreiner1, Schreiner2, Schreiner3, and the Maple Guide, with or without the Distributed Maple Code, renders claims 1, 6, 10, 12, 13, 15, and 29 obvious. Accordingly, Petitioner establishes a reasonable likelihood of prevailing with respect to claims 1, 6, 10, 12, 13, 15, and 29.

E. Alleged Obviousness of Claims 4, 5, 7, and 8 over Schreiner1, Schreiner2, Schreiner3, Maple Guide, Distributed Maple Code, and Deitel

1. Deitel (Ex. 1019)

Deitel generally teaches implementation of a network operating system (“OS”):

Advances in telecommunications technology have profoundly affected OS's. A **network OS** enables its processes to access resources (e.g., files) that reside on other independent computers on a network. . . . In a networked environment, a process can execute on the computer on which it is created or on another computer on the network. In some network operating systems, users can specify exactly where their processes run; in others, the operating system, determines where processes are executed. For example, the system may determine that a process can be more efficiently executed on a computer experiencing a light load.

IPR2021-00075
Patent 8,140,612 B2

Ex. 1019, 38.

Deitel also describes benefits of partitioning memory space and issues associated therewith:

We (as OS designers) view main memory in terms of memory organization. Do we place only a single process in main memory, or do we place several processes in memory at once (i.e., do we implement multiprogramming)? If main memory contains several processes simultaneously, do we give each the same amount of space, or do we divide main memory into portions (called partitions) of different sizes?

Ex. 1019, 379.

2. *Alleged Obviousness for Claims 4, 5, 7, and 8*

Petitioner contends claims 4, 5, 8, and 8, which depend directly or indirectly from claim 1, would have been obvious over the combination of Schreiner1, Schreiner2, Schreiner3, Maple Guide, Distributed Maple Code, and Deitel. *See* Pet. 59–72. Claim 4 requires an “OS” to “incorporate[] at least one of the plurality of cluster node modules.” Claim 5 depends from claim 4 and requires the OS to include “a programming interface exposed to an application program, the programming interface being configured to allow the application program to access functionality supplied by the at least one cluster node module.”

Petitioner relies on Deitel to suggest moving some of the Distributed Maple functions into an OS, and to provide an application interface to utilize some of these functions. *See* Pet. 62–68. Petitioner contends it would have been obvious to employ well-known application interfaces (“APIs”) in an OS in order to aid a user via mouse clicks or windows in simplifying access to functions in Distributed Maple. *See id.*

IPR2021-00075
Patent 8,140,612 B2

Claims 7 and 8 generally and respectively require single-node kernels and cluster node modules to be “configured to access a partitioned space to perform processing of data.” Relying on Deitel, Petitioner contends with respect to claim 7 that it would have been obvious to partition the memory as claimed, because “[i]t was common in the prior art to include partitions in memory for the OS space and the user space so that user applications could not interfere with the OS.” *See* Pet. 69 (“A process can interfere with the OS’s memory—either intentionally or inadvertently—by replacing some or all of its memory contents with other data. . . . Protection in a single-user contiguous memory allocation systems can be implemented with a single boundary register built into the processor.” (quoting Ex. 1019, 387–3880); citing Ex. 1005 ¶ 151)). Petitioner sets forth a similar showing with respect to claim 8. *See id.* at 71–72.

Based on the foregoing discussion and a review of the record, for purposes of institution, Petitioner sufficiently shows that the combination of Schreiner1, Schreiner2, Schreiner3, Maple Guide, Distributed Maple Code, and Deitel renders obvious the subject matter of dependent claims 4, 5, 7, and 8. Patent Owner does not address the dependent claims separately. Nonetheless, the burden remains on Petitioner to demonstrate unpatentability. *See Dynamic Drinkware, LLC v. Nat’l Graphics, Inc.*, 800 F.3d 1375, 1378 (Fed. Cir. 2015)

Accordingly, Petitioner establishes a reasonable likelihood of prevailing with respect to claims 4, 5, 7, 8.

IPR2021-00075
 Patent 8,140,612 B2

F. Alleged Obviousness of Claims 4, 5, 7, and 8 over Schreiner1, Schreiner2, Schreiner3, Maple Guide, Distributed Maple Code, Deitel, and Chen

Petitioner alternatively adds Chen to its showing of alleged obviousness of claims 4 and 5 as described in the previous section. *See* Pet. 72–75. According to Petitioner, “Chen taught that integrating Distributed Maple’s fault tolerance mechanisms implemented in the scheduler with the OS would increase fault tolerance.” *Id.* at 73–74 (citing Ex. 1020, 75–76). Petitioner also contends that “integrating the scheduler in kernel space would allow a significant improvement in fault tolerance, and, as a further benefit, eliminate the need for sending copies of task results to the root node described in Schreiner3, resulting in an even more efficient system.” *Id.* at 74 (citing Ex. 1020, 76, 81; Ex. 1005 ¶ 158).

Based on the foregoing discussion and a review of the record, for purposes of institution, Petitioner sufficiently shows that the combination of Schreiner1, Schreiner2, Schreiner3, Maple Guide, Distributed Maple Code, Deitel, and Chen renders obvious the subject matter of dependent claims 4 and 5.¹¹ Patent Owner does not address the dependent claims separately as

¹¹ Petitioner erroneously alleges that claims 7 and 8 depend from claim 4 and relies on that alleged dependency to assert the obviousness of claims 7 and 8 based further on Chen. Pet. 75. Therefore, Petitioner’s showing based on Chen does not address claims 7 and 8 with sufficient particularity in this alternative ground of rejection. *See* Pet. 72–75. Nonetheless, the Board institutes on all challenges, and Petitioner provides sufficient particularity with respect to claims 7 and 8 in its alternative ground based on Deitel as addressed above (§ II.E). *See SAS Inst. Inc. v. Iancu*, 138 S. Ct. 1348, 1355 (2018) (“The Director, we see, is given only the choice ‘whether’ to institute an inter partes review. That language indicates a binary choice—either institute review or don’t.”).

IPR2021-00075
Patent 8,140,612 B2

noted above. Nonetheless, the burden remains on Petitioner to demonstrate unpatentability. *See Dynamic Drinkware*, 800 F.3d at 1378.

G. The Petition's Word Limit

Patent Owner alleges that the Petition “undercount[s] the true word count” by “about 320 excess words” because it uses “non-standard formats such as ‘EX-1001,’ ‘Schreiner1,’ and ‘¶216’ to make two words count as one and cop[ies] text from the alleged prior art as images on pages 23–24, 27, and 50.” Prelim. Resp. 55 (citing *Starbucks Corp. v. Ameranth, Inc.*, CBM2015-00091, Paper 16 at 2–3 (PTAB Jan. 29, 2016)). It is not clear if by “320 excess words,” Patent Owner implies that the total word count would be 14,320, or if it would be 320 words in excess of the certified word count of 13,882. *See* Prelim. Resp. 55 (noting “37 C.F.R. § 42.24(a)(i) limits the Petition to 14,000 words” and “[t]he Petition certifies its word count is 13,882”).

In any case, the allegedly non-standard formats and copied text from images do not result in an excessive word count or violate the spirit of the word count underlying the rule. For example, the Petition repeats the same annotated Figure 1 on pages 23, 28, and 33, but there are no figures on pages 24, 27, and 50. Petitioner adds words to the same annotated figure on pages 18, 23, 28, 33, and 38 of the Petition to identify how Petitioner reads elements of claim 1 onto the figure, resulting in clarity and ease of reading. However, Petitioner simply could have refrained from reproducing the same figure multiple times in order to minimize the word count at the expense of clarity and ease of reading. Patent Owner does not cite a case in which the Board granted relief based on the use of such alleged non-standard formats for citations. Patent Owner also does not specify any form of relief that it

IPR2021-00075
Patent 8,140,612 B2

seeks. *See id.* Nevertheless, the Board hereby authorizes both parties to use the allegedly non-standard citation formats in future papers in this trial.

III. CONCLUSION

After considering the evidence and arguments presented in the Petition and the Preliminary Response and additional briefing, we determine Petitioner has demonstrated a reasonable likelihood that it would prevail with respect to most of its unpatentability challenges and with respect to the challenged claims. *See supra* note 11. We institute an *inter partes* review on the challenged claims and all of the grounds presented in the Petition. At this stage of the proceeding, we have not made a final determination as to the patentability of these challenged claims.

IV. ORDER

Accordingly, it is

ORDERED that pursuant to 35 U.S.C. § 314, *inter partes* review is instituted as to the challenged claims of the '612 patent with respect to all grounds of unpatentability presented in the Petition; and

FURTHER ORDERED that *inter partes* review is commenced on the entry date of this Order, and pursuant to 35 U.S.C. § 314(c) and 37 C.F.R. § 42.4, notice is hereby given of the institution of a trial.

IPR2021-00075
Patent 8,140,612 B2

For PETITIONER:

Brent Yamashita
Jonathan Hicks
DLA Piper LLP
brent.yamashita@dlapiper.com
jonathan.hicks@dlapiper.com
NVIDIA-ACS-IPR@us.dlapiper.com

For PATENT OWNER:

Jon W. Gurka
Ted M. Cannon
Cheryl T. Burgess
Knobbe Martens Olson and Bear, LLP
2jwg@knobbe.com
2tmc@knobbe.com
2ctb@knobbe.com
BoxZTANNL.017LP@knobbe.com

Exhibit 4

Trials@uspto.gov
571-272-7822

Paper 9
Entered: May 13, 2021

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

NVIDIA CORP.,
Petitioner,

v.

ADVANCED CLUSTER SYSTEMS, INC.,
Patent Owner.

IPR2021-00108
Patent 8,676,877 B2

Before KARL D. EASTHOM, ARTHUR M. PESLAK, and
SEAN P. O'HANLON, *Administrative Patent Judges*.

EASTHOM, *Administrative Patent Judge*.

DECISION
Granting Institution of *Inter Partes* Review
35 U.S.C. § 314

IPR2021-00108
Patent 8,676,877 B2

NVIDIA Corp. (“Petitioner”) filed a Petition (Paper 1, “Pet.”) requesting an *inter partes* review of claims 1, 5, and 8–11 (the “challenged claims”) of U.S. Patent No. 8,676,877 B2 (Ex. 1001, the “’877 patent”). Advanced Cluster Systems, Inc. (“Patent Owner”) filed a Preliminary Response (Paper 5, “Prelim. Resp.”). Pursuant to the Board’s authorization (Paper 6), the parties filed additional briefing. Paper 7 (“Pet. Reply”); Paper 8 (“PO Sur-reply”).

The Board has authority to determine whether to institute an *inter partes* review. See 35 U.S.C. § 314(b); 37 C.F.R. § 42.4(a) (2020). Under 35 U.S.C. § 314(a), authorization of an *inter partes* review requires the information in the Petition and the Preliminary Response to “show[] that there is a reasonable likelihood that the petitioner would prevail with respect to at least 1 of the claims challenged in the petition.” For the reasons that follow, we institute an *inter partes* review as to the challenged claims of the ’877 patent on all grounds of unpatentability presented.

I. BACKGROUND

A. *Real Parties-in-Interest*

Petitioner identifies itself as the real party-in-interest. Pet. 2. Patent Owner identifies itself as the real party-in-interest. Paper 3, 1.

B. *Related Proceedings*

The parties indicate that the ’877 patent is the subject of *Advanced Cluster Systems, Inc. v. NVIDIA Corp.*, No. 19-cv-02032 (D. Del. filed Oct. 28, 2019). Pet. 3; Paper 3, 1. The parties collectively identify the following *inter partes* review petitions filed against patents related to the ’877 patent: IPR2020-01608, IPR2021-00019, IPR2021-00020, and IPR2021-00075. Pet. 3;

IPR2021-00108
Patent 8,676,877 B2

Paper 3, 1.

C. The '877 patent

The '877 patent, titled “Cluster Computing Using Special Purpose Microprocessors,” “relates to the field of cluster computing generally and to systems and methods for adding cluster computing functionality to a computer program, in particular.” Ex. 1001, code (54), 1:22–24.

Embodiments of the '877 patent include enabling a software package to benefit from a plurality of nodes in a cluster. Ex. 1001, 2:12–24. For example, “[o]ne embodiment adapts a software module designed to run on a single node, such as, for example, the Mathematica kernel, to support cluster computing, even when the software module is not designed to provide such support.” *Id.* at 2:24–27. “One embodiment provides parallelization for an application program, even if no access to the program's source code is available.” *Id.* at 2:28–30. “One embodiment provides access to [] high-performance computing through a Mathematica Front End, a command line interface, one or more high-level commands, or a programming language such as C or FORTRAN.” *Id.* at 2:21–24.

IPR2021-00108
 Patent 8,676,877 B2

Figure 1 of the '877 patent follows:

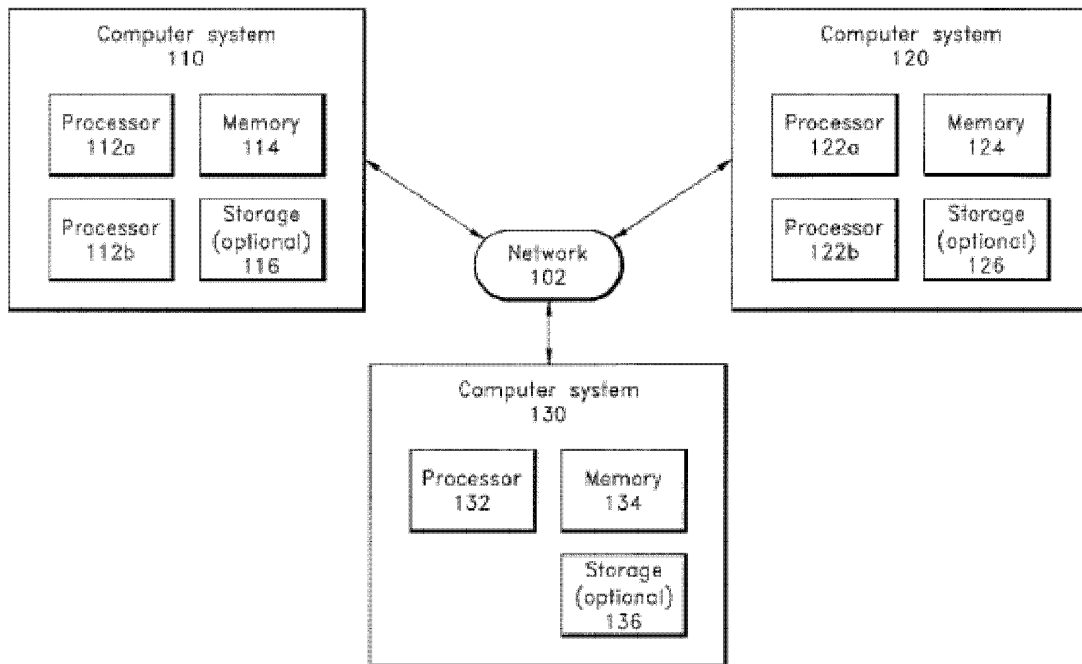


FIG. 1

Figure 1 “is a block diagram of an embodiment of a computer cluster 100 wherein computer systems 110, 120, 130 communicate with one another via a communications network 102.” Ex. 1001, 4:65–5:1. Each computer system includes at least one processor 112a, 112b, 122a, 122b, 132, memory 114, 124, 134, and, optionally, storage 116, 126, 136. *See id.* at 5:1–32. Each processor includes an independent processing core, or “node,” that is capable of single-threaded execution. *See id.* at 4:48–50, 5:8–11.

IPR2021-00108
 Patent 8,676,877 B2

Figure 2 of the '877 patent, showing relationships among software modules in computer cluster 100 (Ex. 1001, 5:18–20), follows:

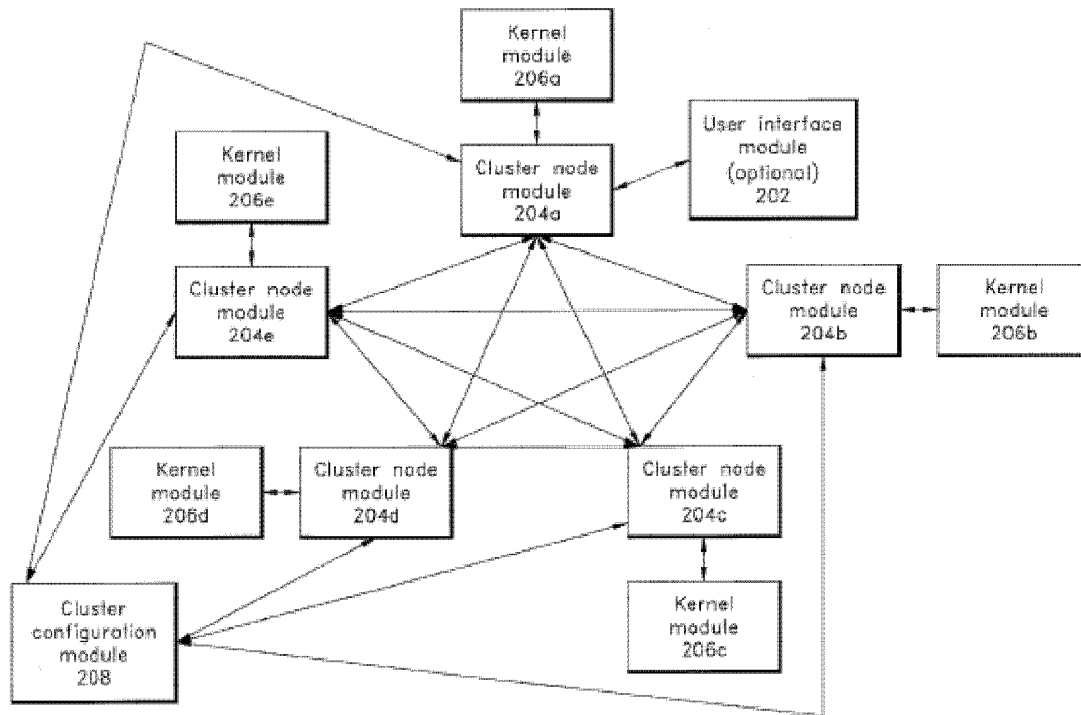


FIG. 2

Figure 2 above illustrates user interface module 202 in communication with cluster node module 204a, which in turn, is in communication with its kernel module 206a, cluster configuration module 208, and other cluster node modules 204b, 204c, 204d, and 204e, with each cluster node module respectively in communication with kernel modules 206b, 206c, 206d, and 206e. *See* Ex. 1001, 5:34–55.

In the embodiment of Figure 2, each kernel module communicates with a single cluster node module. *See* Ex. 1001, 5:34–38. Cluster node modules are software modules that “can include” at least a portion of the message-passing interface (MPI) application programming interface (API) to interact with an application, such as Mathematica. *See id.* at 11:34–53. In

IPR2021-00108
 Patent 8,676,877 B2

the Figure 2 embodiment, each cluster node module communicates with each of the other cluster node modules. *Id.* at 5:42–55. Also, as indicated above, one of the cluster node modules (module 204a) communicates with user interface module 202. *Id.* at 11:6–15. That cluster node module receives commands from the user interface and submits the commands to all of the other cluster node modules. *Id.* at 23:2–8, 23:44–50, 24:11–18. Each cluster node module communicates the command to its respective kernel module, such as for example, Mathematica kernels. *Id.* at 23:18–23, 24:27–33. Each kernel module processes the command and returns a result to its respective cluster node module. *Id.* at 24:34–38. The cluster node module can report the result to the other cluster node modules. *Id.* at 24:38–57. This “interact[ion] on a pair-wise or collective basis” allows code running within multiple, simultaneously running kernel modules to perform calculations, processing, or other work on a larger scale and faster than one kernel acting alone. *Id.* at 24:62–67.

An “interpreter,” “sometimes called a kernel,” “executes instructions provided to the program by a user, a script, or another source” and “can manage at least some hardware resources of a computer system and/or can manage communications between those resources and software (for example, the provided instructions, which can include a high-level programming language).” Ex. 1001, 1:37–46.

D. Illustrative Claims 1 and 10

The Petition challenges claims 1, 5, and 8–11. Pet. 1. Claims 1, 8 and 10 are independent. Independent claims 1 and 10 illustrate the challenged claims at issue:

IPR2021-00108

Patent 8,676,877 B2

1. [a.][i.] A system for performing an instruction received from a front end [ii] by executing commands [iii.] on one or more special purpose microprocessors, the system comprising:

[b.][i.] a plurality of nodes, wherein each node is configured to access a computer-readable memory system comprising [ii.] program code for a single-node kernel module, and wherein each single-node kernel module is [iii.] configured to interpret instructions received by the single-node kernel module into commands that are executable by a special purpose microprocessor;

[c.][i.] a plurality of cluster node modules, wherein [ii.] each cluster node module is stored in a computer-readable memory system and [iii.] configured to communicate with a single-node kernel and with one or more other cluster node modules, [iv.] to accept instructions, and to interpret at least some of the instructions such that the plurality of cluster node modules communicate with one another in order to act as a cluster in executing commands using one or more hardware processors; and

[d.] a communications system configured to connect the plurality of nodes;

[e.] wherein the plurality of cluster node modules cooperate to interpret and translate, as needed, the instruction for execution by a plurality of single-node kernel modules, and

[f.] wherein at least one of the plurality of cluster node modules returns a result to the front end.

Ex. 1001, 29:54–30:12 (information added to conform to Petitioner’s nomenclature).

10. [a.] A method of performing an instruction received from a front end by executing commands on one or more special purpose microprocessors, the method comprising:

[b.] communicating an instruction from a front end to one or more cluster node modules connected to one another by a communications system;

IPR2021-00108
Patent 8,676,877 B2

[c.] for each of the one or more cluster node modules, communicating a message based on the instruction to a respective kernel module associated with the cluster node module, wherein the respective kernel module is configured to interpret the message into commands that are executable by a special purpose microprocessor;

[d.] for each of the one or more cluster node modules, receiving a result from the respective kernel module associated with the cluster node module; and

[e.] for at least one of the one or more cluster node modules, responding to messages from other cluster node modules.

Ex. 1001, 30:59–31:9 (information added to conform to Petitioner’s nomenclature).

E. The Asserted Challenges to Patentability

The Petition relies on the following references:

Wolfgang Schreiner et al., *Distributed Maple: Parallel Computer Algebra in Networked Environments*, 35 Journal of Symbolic Computation 305 (2003) (Ex. 1008) (“Schreiner1”);

Wolfgang Schreiner, *Distributed Maple – User and Reference Manual (V 1.1.12)* (2001) (Ex. 1009) (“Schreiner2”);

Károly Bósa and Wolfgang Schreiner, *Task Logging, Rescheduling and Peer Checking in Distributed Maple* (2002) (Ex. 1010) (“Schreiner3”);

K. M. Heal et al., *Maple V Learning Guide* (J. S. Devitt ed., 1998) (Ex. 1011) (“Maple Guide”);

“Distributed Maple Code” references (Ex. 1012–Ex. 1018) (Pet. xii):

Dist.Maple5: “[S]ource code for Distributed Maple” (Ex. 1012);

Maple Function Source Code: “Source code for parallel versions of Maple functions in Distributed Maple from the ‘distsoft’ directory” (Ex. 1013);

IPR2021-00108

Patent 8,676,877 B2

CASA Function Source Code: “Source code for parallel versions of CASA functions in Distributed Maple from the ‘distsoft’ directory” (Ex. 1014);

Install1 File: “‘Install’ file for Distributed Maple” (Ex. 1015);

ReadMe1 File: “‘ReadMe’ file for Distributed Maple” (Ex. 1016);

Install2 File: “‘Install’ file for source code in ‘distsoft’ directory” (Ex. 1017);

ReadMe2 File: “‘ReadMe’ file for source code in ‘distsoft’ directory” (Ex. 1018);

SUN Debuts UltraSPARC IV, EE Times, October 15, 2003 (“SPARC IV Article”) (Ex. 1019);

AMD to Unveil Dual-Core PC Chips, Los Angeles Times, May 31, 2005 (“AMD Article”) (Ex. 1020);

Maple V, PC Magazine, August 1992, 419–425 (“PC Magazine1”) (Ex. 1029);

Maple V Looks Better With Improved Graphics, PC Magazine, July 1993, 50 (“PC Magazine2”) (Ex. 1030);

Anshuman Nayak, et al., “A Library based compiler to execute MATLAB Programs on a Heterogeneous Platform” (“Nayak”) (Ex. 1031);

Kelly et al., U.S. Pat. No. 6,691,216 (filed October 24, 2001, issued Feb. 10, 2004) (“Kelly”) (Ex. 1032).

Petitioner refers to Exhibits 1008–1010 and 1012–1018 collectively as “Distributed Maple Publications.” Pet. 7–8.

Petitioner asserts the following grounds of unpatentability:

Claim(s) Challenged	35 U.S.C. §	Reference(s)/Basis
1, 8–11	103(a) ¹	Schreiner1, Schreiner2, Schreiner3, Maple Guide,

¹ The filing date of the application resulting in the ’877 patent precedes the effective date of Leahy-Smith America Invents Act (“AIA”), Pub. L. No.

IPR2021-00108
Patent 8,676,877 B2

Claim(s) Challenged	35 U.S.C. §	Reference(s)/Basis
		Distributed Maple Code, PC Magazine1, PC Magazine 2
1, 10, 11	103(a)	Schreiner1, Schreiner2, Schreiner3, Maple Guide, Distributed Maple Code, PC Magazine1, PC Magazine2, Nayak
5	103(a)	Schreiner1, Schreiner2, Schreiner3, Maple Guide, Distributed Maple Code, PC Magazine1, PC Magazine2, Nayak, Kelly
5	103(a)	Schreiner1, Schreiner2, Schreiner3, Maple Guide, Distributed Maple Code, PC Magazine1, PC Magazine2, SPARC IV Article, AMD Article

See Pet. 11. Petitioner submits declarations of Henry Tufo, Ph.D. (Ex. 1005) and Wolfgang Schreiner, Ph.D. (Ex. 1006), and Sylvia Hall-Ellis, Ph.D. (Ex. 1007) in support of its contentions. Patent Owner submits declarations of Jaswinder Pal Singh, Ph.D. (Ex. 2001), Vineer Bhansali, Ph.D. (Ex. 2007), John Bancroft (Ex. 2008), and Dean Dager, Ph.D. (Ex. 2036), in support of its Preliminary Response.

II. ANALYSIS

Petitioner challenges claims 1, 5, and 8–11 as obvious based on the grounds listed above. Pet. 1. Patent Owner disagrees. Prelim. Resp. 1–8.

112–29, 125 Stat. 284 (2011). Thus, reference is to the pre-AIA version of section 103.

IPR2021-00108
 Patent 8,676,877 B2

A. Legal Standards

35 U.S.C. § 103(a) renders a claim unpatentable if the differences between the claimed subject matter and the prior art are such that the subject matter, as a whole, would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. *See KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 406 (2007).

Tribunals resolve obviousness based on underlying factual determinations, including (1) the scope and content of the prior art; (2) any differences between the claimed subject matter and the prior art; (3) the level of skill in the art; and (4) where in evidence, so-called secondary considerations. *See Graham v. John Deere Co.*, 383 U.S. 1, 17–18 (1966). Prior art references must be “considered together with the knowledge of one of ordinary skill in the pertinent art.” *In re Paulsen*, 30 F.3d 1475, 1480 (Fed. Cir. 1994) (citing *In re Samour*, 571 F.2d 559, 562 (CCPA 1978)).

B. Level of Ordinary Skill in the Art

Petitioner relies on Dr. Tufo, who testifies that a person having ordinary skill in the art at the time of the invention (“POSITA”) would have had “a Bachelor’s degree in computer science, electrical engineering, or an equivalent field, and two years of academic or industry experience in parallel and/or distributed computing.” Ex. 1005 ¶ 40; Pet. 14 (citing Ex. 1005 ¶¶ 38–41).

On one hand, the Preliminary Response does not present a proposed level of ordinary skill at this preliminary stage. On the other hand, Dr. Singh, Patent Owner’s declarant, disagrees with Dr. Tofu’s proposed level of

IPR2021-00108
 Patent 8,676,877 B2

ordinary skill. Ex. 2001 ¶¶ 33–34.² The Preliminary Response does not cite Dr. Singh’s proposal. For purposes of this Decision on Institution, we adopt Petitioner’s proposed level of ordinary skill in the art, which comports with the teachings of the ’877 patent and the asserted prior art.

C. Claim Construction

In an *inter partes* review, the Board construes each claim “in accordance with the ordinary and customary meaning of such claim as understood by one of ordinary skill in the art and the prosecution history pertaining to the patent.” 37 C.F.R. § 42.100(b) (2020). Under the same standard applied by district courts, claim terms have their plain and ordinary meaning as would have been understood by a person of ordinary skill in the art at the time of the invention and in the context of the entire patent disclosure. *Phillips v. AWH Corp.*, 415 F.3d 1303, 1313 (Fed. Cir. 2005) (en banc). “There are only two exceptions to this general rule: 1) when a patentee sets out a definition and acts as his own lexicographer, or 2) when the patentee disavows the full scope of a claim term either in the specification or during prosecution.” *Thorner v. Sony Comput. Entm’t Am. LLC*, 669 F.3d 1362, 1365 (Fed. Cir. 2012).

² Dr. Singh contends that “‘parallel computing’ is a category that includes ‘cluster computing’ but ‘distributed computing’ is different from both ‘cluster computing’ and ‘parallel computing.’” *Id.* ¶ 33. Accordingly, Dr. Singh testifies that a “POSITA would have had a Bachelor’s degree in computer science, electrical engineering, or an equivalent field, and two years of academic or industry experience in **cluster computing**.” *Id.* ¶ 34. However, Dr. Singh states that “my conclusions would not change even assuming that Dr. Tufo’s identification of the level of ordinary skill in the art were correct.” *Id.* For purposes of institution, Dr. Singh’s testimony reveals that any differences in the proposed level of ordinary skill raised by Dr. Singh are not material.

IPR2021-00108
Patent 8,676,877 B2

Claim 1 recites the following “cluster node module” limitation, which raises one of two claim construction disputes here:

a plurality of cluster node modules, wherein each cluster node module is stored in a computer-readable memory system and configured to communicate with a single-node kernel and with one or more other cluster node modules, to accept instructions, and to interpret at least some of the instructions such that the plurality of cluster node modules communicate with one another in order to act as a cluster in executing commands using one or more hardware processors.

Each challenged claim recites a “cluster node module,” although the challenged claims do not all recite the limitations (as quoted immediately above) that define how each cluster node module is “configured” in claim 1. For example, in addition to reciting receiving and responding limitations (*supra* § I.D), claim 10 also recites the following “cluster node module” limitations, which raises the second claim construction dispute here:

communicating an instruction from a front end to one or more cluster node modules connected to one another by a communications system;

for each of the one or more cluster node modules, communicating a message based on the instruction to a respective kernel module associated with the cluster node module, wherein the respective kernel module is configured to interpret the message into commands that are executable by a special purpose microprocessor.

Generally, “Petitioner believes that no express construction of any term is needed to resolve the challenges herein but reserves the right to present express constructions in response to any argument by P[atent] O[wner].” Pet. 14.³ Notwithstanding Petitioner’s statement, Patent Owner

³ Neither party argues that term “module” as recited in the challenged claims (i.e., “cluster node module” or “kernel module”) is a means-plus-function

IPR2021-00108
 Patent 8,676,877 B2

argues that “[t]he Petition does not comply with the requirement to construe the claims because it does not construe *any* claim terms, even though construction is necessary to resolve the Petition.” Prelim. Resp. 47. To the contrary, Petitioner sufficiently identifies how it reads the claims onto Schreiner1’s system, as discussed below. *See infra* § II.D.8.a. Also, it is evident from Patent Owner’s claim construction arguments (addressed in this section and further below (*id.*)) that Patent Owner disagrees with, and therefore understands, the two central claim construction disputes underlying Petitioner’s implicit claim constructions.

First, Patent Owner notes that each challenged claim recites a “cluster node module.” Prelim. Resp. 9. This raises the first claim construction dispute as indicated above. Specifically, Patent Owner argues that

[t]he Board should construe “cluster node module” to mean “a module that cooperates with other cluster node modules to establish intercommunication among nodes in a computer cluster and to exchange messages such that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.”

Id. at 16 (citing Ex. 2001 ¶ 50).

term even though “‘module’ is a well-known nonce word that can operate as a substitute for ‘means’ in the context of § 112, para. 6.” *See Williamson v. Citrix Online, LLC*, 792 F.3d 1339, 1350 (Fed. Cir. 2015) (en banc) (“use of the word ‘means’ creates a presumption that [35 U.S.C.] § 112, ¶ 6 applies”); *see* 35 U.S.C.] § 112, ¶ 6 (“An element in a claim for a combination may be expressed as a means . . . for performing a specified function without the recital of structure, material, or acts in support thereof, and such claim shall be construed to cover the corresponding structure, material, or acts described in the specification and equivalents thereof.”). Because neither party argues that “module” is a means-plus-function term, we decline to construe it as such for institution purposes.

IPR2021-00108
 Patent 8,676,877 B2

Patent Owner raises the second claim construction dispute with respect to claims 8 and 10.⁴ *See* Prelim. Resp. 16–19. Patent Owner contends that independent claim 10 recites “communicating an instruction *from* a front end *to* one or more cluster node modules” and “for each of the one or more cluster node modules, communicating a message based on the instruction *to* a respective kernel module associated with the cluster node module.” Prelim. Resp. 16 (emphasis by Patent Owner). According to Patent Owner, the Board should construe these recited phrases

as establishing the following relative order in which an instruction is processed by the front end, cluster node modules, and kernels: (1) first, an instruction starts at the front end, (2) second, an instruction is “communicat[ed] [. . .] from [the] front end to” the “cluster node modules,” and, (3) third, a “message based on the instruction” is “communicat[ed] . . . to a respective kernel.

Id. at 19–20 (quoting claim 10) (all alterations except the ellipses by Patent Owner).

Under this “relative order,” Patent Owner argues that claim 10 “does *not* cover a different relative order in which the front end sends instructions to the kernels and the kernels then forward instructions to the cluster node modules.” Prelim. Resp. 17. Patent Owner summarizes its claim construction argument for claim 10 in the following diagram:

⁴ Patent Owner focuses only on claim 10 under this claim construction argument. However, independent claim 8 recites the same relevant limitations as independent claim 10 with respect to this dispute.

IPR2021-00108
Patent 8,676,877 B2

Correct order:	Front End → Cluster Node Module → Kernel
Incorrect order:	Front End → Kernel → Cluster Node Module

The diagram above represents Patent Owner’s claim construction wherein Patent Owner labels as an “[i]ncorrect order” a construction of claim 10 that allows a kernel to pass an instruction from a front end to a cluster node module. *See id.* at 16–17 (citing Ex. 2001 ¶ 51). In other words, Patent Owner argues that claim 10 “excludes the kernels receiving instructions from the front end and forwarding instructions to the cluster node modules.” *Id.* at 19–20. However, Patent Owner clarifies that the instructions can pass indirectly through *any* “device[]” or “component[]” *except a kernel*: “To be clear, this construction does **not** require instructions to be transmitted directly from the front end to the cluster node modules without passing through other devices, such as routers, switches, or other components.” *See id.* at 20 n.4.

Therefore, both claim construction disputes involve assertions of implied negative limitations with respect to the cluster node modules as recited in the challenged claims. Contrary to Patent Owner’s arguments, the plain language of claim 1 does not preclude indirect communications between cluster node modules “through a central server or master node.” Similarly, the plain language of claim 10 does not preclude instructions from passing indirectly through a “kernel” before they reach the cluster node modules.

Rather, claim 1 recites “each cluster node module is . . . configured to communicate with a single-node kernel and with one or more other cluster node modules, to accept instructions, and to interpret at least some of the

IPR2021-00108
Patent 8,676,877 B2

user instructions such that the plurality of cluster node modules communicate with one another in order to act as a cluster.” This language requires “each cluster node module . . . to communicate with a single-node kernel” without specifying anything about a central server or master node. The language also requires “each cluster node module . . . to accept instructions, and to interpret at least some of the instructions such that the plurality of cluster node modules communicate with one another in order to act as a cluster in executing commands using one or more hardware processors.” This language does not prevent the communications from passing through a central server or master node.

Similar remarks apply to the plain language of claim 10. As Patent Owner notes, claim 10 first recites “communicating an instruction from a front end to one or more cluster node modules” and then recites “for each of the one or more cluster node modules, communicating a message based on the instruction to a respective kernel module associated with the cluster node module.” *See* Prelim. Resp. 16. But these limitations simply do not recite the proposed negative limitation “without passing through a kernel.” The first “communicating” limitation recites “communicating an instruction from a front end to one or more cluster node modules” without mentioning a kernel and without mentioning a direct connection. Just because the cluster node module “communicat[es] a message based on the instruction [from a front end] to a respective kernel module associated with the cluster node module,” this does not preclude the instruction from passing through a kernel or any other device or component on its way from a user interface to one or more cluster node modules. As noted above, Patent Owner generally argues that claim 10 allows an instruction to pass through intervening

IPR2021-00108
 Patent 8,676,877 B2

“devices” or “other components” on its way from a user interface to one or more cluster node modules. Prelim. Resp. 20 n.4.

Contrary to Patent Owner’s other arguments, the specification does not limit the claims in the manner argued. *See* Prelim. Resp. 13 (citing Ex. 1001, 22:45–46, 23:12–14, Fig. 2, Ex. 1001 ¶ 47). Patent Owner relies on Figure 2 and other selected passages to incorporate limitations into “cluster node modules” as recited in each challenged claim (*see id.*), but the specification states that “[t]he drawings and the associated descriptions are provided to illustrate embodiments and *not to limit the scope of the disclosure.*” Ex. 1001, 3:65–67 (emphasis added). Moreover, following guidance from the Federal Circuit, generally “[w]e do not read limitations from the embodiments in the specification into the claims.” *Hill-Rom Servs., Inc. v. Stryker Corp.*, 755 F.3d 1367, 1371 (Fed. Cir. 2014) (citing *Liebel-Flarsheim Co. v. Medrad, Inc.*, 358 F.3d 898, 904 (Fed. Cir. 2004)).

Similarly, the specification describes several embodiments under a “SUMMARY” of the invention section in general terms “[w]ithout limiting the scope of the invention.” *See* Ex. 1001, 2:7–11. One passage mimics the broad language of claims 8 and 10, which plainly does not provide the negative limitations argued by Patent Owner:

Each cluster node module is configured to communicate with a single node kernel and with one or more other cluster node modules, to accept instructions from the user interface, and to interpret at least some of the user instructions such that the plurality of cluster node modules communicate with one another in order to act as a cluster.

Id. at 3:24–30.

This generic passage allows each cluster node module “to accept instructions from the user interface,” without requiring such a module to

IPR2021-00108
Patent 8,676,877 B2

accept them directly from the user interface. The passage says nothing about precluding passage of a message through a master node, central server, or kernel.

Patent Owner also relies on Figure 2 and the specification as showing “‘direct connections’ between each cluster node module” in a peer-to-peer architecture. Prelim. Resp. 13 (citing Ex. 1001, 22:45–46, 23:12–14; Ex. 2001 ¶ 47). Patent Owner similarly contends that “there is no connection between the user interface module 202 and the kernel module 206a or any of the other kernel modules.” *Id.* at 19. At one of the cited passages, the specification states “[t]he cluster node modules 204a–e establish communication with one another at 404. *In one embodiment*, each of the cluster node modules 204a–e establish[es] *direct connections* . . . with other cluster node modules . . . *launched on the computer cluster 100 by the cluster configuration module 208.*” Ex. 1001, 23:12–17 (emphasis added).

As noted above, however, embodiments in the specification generally do not limit the claim terms. *See Hill-Rom Servs.*, 755 F.3d at 1371. Moreover, the cited passage implies that any direct connection only applies to “one embodiment.” Ex. 1001, 23:12–17. Further, that embodiment includes “cluster configuration module 208.” *See id.* But Patent Owner specifically argues that such modules “are optional parts of the cluster node modules,” because the specification refers to “one embodiment.” *See* Prelim. Resp. 10–11. Patent Owner also argues other claims in other patents specifically recite “optional parts of the cluster node modules,” further showing that these components are optional. *Id.* at 11 (citing “IPR2020-01608, Ex. 1001, at claim 8 (received message queue), claim 9 (message

IPR2021-00108
 Patent 8,676,877 B2

receiving queue), claim 13 (advanced functions module), claim 26 (MPI module)’’).

Patent Owner’s logic extends to claims in other patents that specifically recite “peer-to-peer” architecture. *See* IPR2021-00019, Ex. 1001, 30:33–35 (claim 1 reciting “a mechanism for the nodes to communicate . . . with each other using a peer-to-peer architecture”). In other words, Patent Owner’s arguments that the disclosed “peer-to-peer” architecture precludes an indirect connection between “cluster modules” in the challenged claims (*see* Prelim. Resp. 12–14) contradict its other arguments that unclaimed features here that other patent claims specifically recite imply the features are optional (*id.* at 10–11).

The specification also contradicts Patent Owner’s arguments on this preliminary record with respect to claim 10. *See* Prelim. Resp. 16–20. For example, it states “[a] kernel module 206 typically includes program code for interpreting high-level code, commands, and/or instructions *supplied by a user or a script* into low-level code, such as, for example, machine language or assembly language.” Ex. 1001, 22:41–44 (emphasis added). Similarly, the specification specifically states that “[t]he operating system or its components *can connect the front end 202 and kernels 206 to one another in the same manner as a cluster node module 204 would or by a variation of one of the techniques described previously.*” *Id.* at 25:30–34 (emphasis added). In other words, contrary to Patent Owner’s arguments, on this preliminary record, the specification supports connecting a user interface directly to any kernel. This connection between a kernel and a user interface implies that instructions pass through a kernel from a user interface to a cluster node module.

IPR2021-00108
 Patent 8,676,877 B2

In line with the above teachings, the specification also describes user interface module 202 communicating with kernel module 202 for “some embodiments” as follows:

In some embodiments, computer cluster 100 includes a user interface module 202, such as, for example a Mathematica Front End or a command line interface, that includes program code for a kernel module 206 to provide graphical output, accept graphical input, and provide other methods of user communication that a graphical user interface or a command-line interface provides. To support a user interface module 202, the behavior of a cluster node module 204a is altered in some embodiments. Rather than sending output to and accepting input from the user directly, the user interface module 202 activates the cluster node module 204a to which it is connected and specifies parameters to form a connection, such as a MathLink connection, between the cluster node module 204a and the user interface module 202. The user interface module’s activation of the cluster node module 204a can initiate the execution of instructions to activate the remaining cluster node modules 204b–e on the cluster and to complete the sequence to start all kernel modules 206a–e on the cluster. Packets from the user interface module 202, normally intended for a kernel module 206a, are accepted by the cluster node module 204a as a user command. Output from the kernel module 206a associated with the cluster node module 204a can be forwarded back to the user interface module 202 for display to a user. Any of the cluster node modules 204a–e can be configured to communicate with a user interface module 202.

Ex. 1001, 22:14–39 (emphasis added).

Patent Owner argues that portions of this passage support its construction of claim 10. Prelim. Resp. 17 (citing Ex. 1001, 22:14–16, 22:32–34). However, in context, the passage, read in its entirety, does not support Patent Owner’s narrow claim construction on this preliminary record. Rather, the first emphasized portion as quoted above explicitly

IPR2021-00108
 Patent 8,676,877 B2

describes communication between user interface module 202 and kernel module 206. *See* Ex. 1001, 22:14–39. It also describes “alter[ing]” “the *behavior* of cluster node module 204a . . . [but only] *in some embodiments*” so that the cluster node module “can initiate the execution of instructions to activate the remaining cluster node modules 204b–e on the cluster and to complete the sequence to start all kernel modules 206a–e on the cluster.” *See id.* (emphasis added). In other words, to the extent altering this “behavior” involving cluster node modules somehow limits behavior with respect to normal kernel communications, it only occurs for “some embodiments”—i.e., a subset of “some embodiments” introduced at the beginning of the passage. *See id.*

In addition, the passage verifies that packets *normally* pass to kernel module 206a (at least for some contemplated embodiments). *See* Ex. 1001, 22:14–39. Therefore, it is only in a subset of the embodiments that packet messages pass from user interface module 202 first, then through cluster node module 204, and finally to kernel module 206a. *See id.* Accordingly, nothing in this passage limits the claims to a direct connection between a user interface and a cluster node module or between cluster node modules, by precluding an intervening kernel, master node, or server. Patent Owner’s claim construction attempts to allow some forms of indirect communication between cluster nodes, by only attempting to preclude an intervening “central server” or “master node,” while otherwise generally allowing for other intervening “components” or “devices,” lacks requisite support in the specification. *See* Prelim. Resp. 20 n.4.

In addition, cluster node module 204a appears to act as a “master node” or “central server” when it is connected to the user interface module,

IPR2021-00108
 Patent 8,676,877 B2

because messages from other kernels (nodes) must pass through cluster node module 204a on their way to the kernel associated with cluster node module 204a and/or to the user interface node.⁵ *See, e.g.*, Ex. 1001, Fig. 2, 6:19–23 (“Results of evaluations performed by kernel modules 206a-e are communicated back to the first cluster node module 204a via the cluster node modules 204a-e, which communicates them to the user interface module 208.”), 11:34–37 (“In one embodiment, the cluster node modules 204a-e provide a way for many kernel modules 206a-e such as, for example, Mathematica kernels, running on a computer cluster 100 to communicate with one another.”), 23:55–57 (“The cluster node module creates an illusion that a kernel module is communicating directly with the other kernel modules.”). Cluster node module 204a also acts as a “central server,” because it instigates connections to the remaining cluster node modules, according to the column 22 passage discussed and reproduced above. It also controls the other cluster node modules in a “procedure to shut down the system.” *See id.* at 25:46–47.

The specification also indicates that a load balancing embodiment includes a “root processor” that assigns tasks to each of the cluster nodes. *See* Ex. 1001, 21:3–12. On this preliminary record, this further shows that the challenged claims do not require a cluster node module “to exchange messages such that each node can communicate tasks and data with other

⁵ According to the specification, “[t]he term ‘node’ refers to a processing unit or subunit that is capable of single-threaded execution of code.” Ex. 1001, 4:48–50. The specification also describes “computers, microprocessors, and/or processor cores (‘nodes’).” *Id.* at 1:26–27.

IPR2021-00108
Patent 8,676,877 B2

nodes *without the tasks and data being required to go through a central server or master node*” as asserted by Patent Owner.

Finally, on this preliminary record, the prosecution history also does not support the negative limitations argued by Patent Owner. *See* Ex. 1002, 71 (“Applicant hereby rescinds and retracts such disclaimer” arising out of “any prior amendments or characterizations of the scope of any claim or referenced art.”).

Based on the foregoing discussion, requisite disclaimer, disavowal, or lexicography does not appear to exist on this preliminary record to import Patent Owner’s proposed negative limitations into the plain language of independent claims 1 and 10. *See Omega Eng’g, Inc. v. Raytek Corp.*, 334 F.3d 1314, 1321–23 (Fed. Cir. 2003) (holding that a claim limitation to “strike the periphery . . . for visibly outlining” an energy zone with a laser does not justify a negative limitation of “not striking the center or interior portion” of the energy zone absent an “express disclaimer or independent lexicography in the written description that would justify adding that negative limitation”).

No other terms require an express construction. Only those terms that are in controversy need be construed, and only to the extent necessary to resolve the controversy. *Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co.*, 868 F.3d 1013, 1017 (Fed. Cir. 2017) (citing *Vivid Techs., Inc. v. Am. Sci. & Eng’g, Inc.*, 200 F.3d 795, 803 (Fed. Cir. 1999)).

D. Alleged Obviousness of Claims 1 and 8–11

Petitioner relies on the combined teachings of Schreiner¹, Schreiner², Schreiner³, the Maple Guide, the Distributed Maple Code, PC Magazine¹, and PC Magazine² to allege obviousness of claims 1 and 8–11. *See* Pet. 11.

IPR2021-00108
Patent 8,676,877 B2

1. *Schreiner1 (Ex. 1008)*

Schreiner1, entitled “Distributed Maple: parallel computer algebra in networked environments,” bears a copyright date of 2003. Ex. 1008, 3.⁶ Schreiner1 is a journal article authored by Dr. Schreiner, Christian Mittermaier, and Karoly Bosa. *Id.* Schreiner1 “gives a comprehensive overview on the design and the use of ‘Distributed Maple,’ an environment for parallel computer algebra on multiprocessors and heterogeneous computer clusters.” *Id.* Schreiner1 explains that Distributed Maple was developed on the basis of the computer algebra system Maple (*id.*) and that Distributed Maple is built on top of the Maple kernel and does not require any kernel extensions. *Id.* at 4. Schreiner1 states that Distributed Maple is “so portable that applications can be executed in many different environments” and “so general that it can be applied to schedule tasks of other computer algebra systems (e.g., Mathematica).” *Id.* Schreiner1 describes Distributed Maple as providing “a programming model which is based on functional/logic/dataflow parallelism” that “allows the creation of a large number of implicitly scheduled tasks with automatic resolution of data dependencies and of globally shared data structures with implicit synchronization.” *Id.*

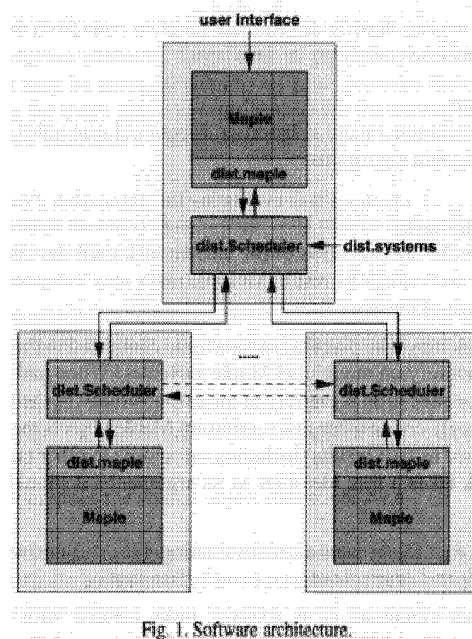
Schreiner1 describes using Distributed Maple “to develop the first parallel versions for a number of non-trivial applications from algebraic

⁶ All page number citations herein refer to page numbers added to the Exhibits by Petitioner. In some cases (for example with respect to Ex. 1008, and Ex. 1011), the Board converted citations by Petitioner to original page numbers into the page numbers added by Petitioner. The Board urges the parties to cite to the page numbers added by the parties to the Exhibits in future briefing.

IPR2021-00108
 Patent 8,676,877 B2

geometry (parallel curve and surface plotting and parallel neighborhood analysis).” Ex. 1008, 5. Schreiner1 discloses that “[t]he user interacts with Distributed Maple via a conventional Maple frontend (text or graphical).” *Id.* at 7. Schreiner1 explains that “[t]he core of Distributed Maple is a scheduler program which is completely independent and even unaware of Maple” and “can in fact embed and schedule tasks from any kind of computation kernels that implement a specific communication protocol.” *Id.* at 8.

Schreiner1 discloses that a Distributed Maple session comprises two main components: a scheduler and a Maple interface. Ex. 1008, 8. Figure 1 of Schreiner1, reproduced below, depicts these components and corresponding software architecture for a Distributed Maple session. *Id.* at 9.



As shown in Figure 1, a Distributed Maple session “comprises a set of nodes each of which holds a pair of processes: a *kernel* and a *scheduler*.”

IPR2021-00108
 Patent 8,676,877 B2

Ex. 1008, 17. “Initially, a single task runs on the root kernel; this task may subsequently create new tasks which are distributed via the schedulers to other kernels and may in turn create new tasks.” *Id.* With reference to Figure 1, Schreiner1 explains that “every scheduler instance accepts tasks from the attached computation kernel and schedules these tasks among all machines connected to the session.” *Id.* at 9. Schreiner1 further explains that “[t]he Maple kernel is a single-threaded process which communicates by a simple communication protocol with the schedule on the same node” and “[a]ll capabilities for parallel and distributed program execution are embedded in this scheduler.” *Id.* at 12.

In general, “[t]he system embeds kernels of the computer algebra system Maple as computational engines into a networked coordination layer implemented in the programming language Java.” Ex. 1008, 3 (Abstract). Maple is a computer algebra system similar to that of Mathematica. *See id.* at 4. The Distributed Maple system “connects external computation kernels on various machines and schedules concurrent tasks for execution on them.” *Id.*

Using “a comparatively high-level programming model, one may write parallel Maple programs that show good speedups in medium-scaled environments.” Ex. 1008, 3 (Abstract). Schreiner1 states “[b]oth the Distributed Maple system itself and the library of parallel version of . . . Maple algorithms are in stable versions freely available under the GUN Library General Public License at <http://www.risc.uni-linz.ac.at/software/distmaple>.” *Id.* at 5.

IPR2021-00108
Patent 8,676,877 B2

2. *Schreiner1 (Ex. 1009)*

Schreiner2, titled “Distributed Maple – User and Reference Manual (V 1.1.12),” by Dr. Schreiner, published on July 6, 2001. Ex. 1009, 1. Schreiner2 describes the same Maple and Distributed Maple system as Schreiner1 in further detail, including programming and system details. *See, e.g., id.* at 1, 4, 13–19. More particularly, Schreiner2 “describes the use of a system for writing distributed Maple applications and sketches its implementation.” *Id.* at 4. Schreiner2 states that its “goal is to provide an environment that makes it easy to implement parallel algorithms in Maple and that can be easily installed in any environment in order to facilitate the distribution of these implementations.” *Id.* As in Schreiner1, Schreiner2 states “[t]he system . . . can be freely downloaded from <http://www.risc.uni-linz.ac.at/software/distmaple>.” *Id.* at 1.

3. *Schreiner3 (Ex. 1010)*

Schreiner3, titled “Task Logging, Rescheduling and Peer Checking in Distributed Maple,” by Dr. Schreiner and others, published on March 18, 2002. Ex. 1010, 1. Schreiner3 describes the same Maple and Distributed Maple system as Schreiner1 in further detail, including system details and fault analysis. *See id.* at 1–14 (also citing <http://www.risc.uni-linz.ac.at/software/distmaple>).

IPR2021-00108
 Patent 8,676,877 B2

Figure 1 of Schreiner3 illustrates an Execution Model of the Distributed Maple system:

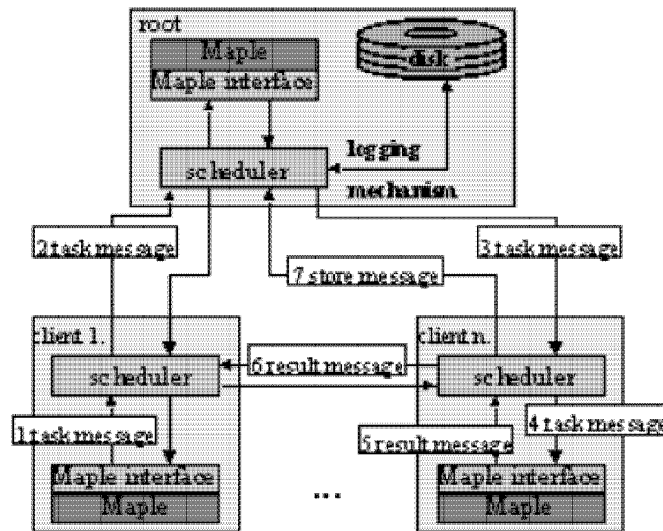


Figure 1: Execution Model

Figure 1 depicts the passing of messages between nodes in a Distributed Maple system, and a logging mechanism in the root node. Ex. 1010, 5.

Schreiner3 explains that “[t]he logging mechanism in Distributed Maple is a fault tolerance mechanism for saving the results of intermediate tasks and the values of shared objects during the computation” and allows the system “to restore the results of computed tasks in a later session, if the current session crashes.” Ex. 1010, 3.

4. Maple Guide (Ex. 1011)

Maple Guide, entitled “Maple V Learning Guide, Release 5,” and published by Waterloo Maple, Inc., bears a copyright date of 1998. Ex. 1011, 5. Maple Guide explains that “Maple V is a *Symbolic Computation System* or *Computer Algebra System*” and that “[b]oth phrases refer to Maple V’s ability to manipulate information in a symbolic or

IPR2021-00108
 Patent 8,676,877 B2

algebraic manner.” *Id.* at 11. Maple Guide is an introductory guide that describes how to use Maple’s graphical interface and the Maple programming language. *Id.* at 13.

5. *Distributed Maple Code (Ex. 1012–Ex. 1018)*

Distributed Maple Code is a collection of the following computer code files embodying a distribution of Distributed Maple for installation on a computer: dist.maple5 file (Ex. 1012), source code for parallel versions of Maple functions in the distsoft directory (Ex. 1013), source code for parallel versions of CASA functions in the distsoft directory (Ex. 1014), an “Install” file for Distributed Maple (Ex. 1015), a “ReadMe” file for Distributed Maple (Ex. 1016), an “Install” file for the source code in the distsoft directory (Ex. 1017), and a “ReadMe” file for the source code in the distsoft directory (Ex. 1018).

6. *PC Magazine1 (Ex. 1029)*

PC Magazine1 is a trade journal article describing aspects of Maple V, by Waterloo Maple Software. It describes an 80387 coprocessor, and states that “Maple V includes libraries for two- and three-dimensional geometry, projective geometry, differential forms, group, theory, Boolean algebra, number theory,” and other mathematical functions. *See* Ex. 1029, 7 (Libraries, Fact File).

7. *PC Magazine2 (Ex. 1030)*

PC Magazine2 is a trade journal article describing aspects of Maple V, by Waterloo Maple Software, including its specialized mathematical functions and graphics. *See* Ex. 1030, 6.

IPR2021-00108
Patent 8,676,877 B2

8. *Claims 1 and 8–11*

a. Analysis of Claims 1 and 8–11

Petitioner relies on the combined teachings of Schreiner1, Schreiner2, Schreiner3, the Maple Guide, the Distributed Maple Code, PC Magazine1, and PC Magazine2, as supported by the testimony of Dr. Tufo and Dr. Schreiner, to allege obviousness of claims 1 and 8–11. *See* Pet. 15–70.

As motivation to combine the “Distributed Maple Publications,” Petitioner contends that they share the same author, Dr. Schreiner, and all relate to the same software project, called “Distributed Maple.” Pet. 15. Petitioner essentially contends that a person of ordinary skill would have consulted the references to learn details about the Maple system, including fault tolerances and capabilities, in order to combine desired features for running the software modules and system. *See id.* at 15–16.

Regarding the Maple Guide (not authored by Dr. Schreiner), according to Petitioner, “Schreiner1 teaches that Distributed Maple includes Maple software modules and refers readers to www.maplesoft.com, a website operated by Waterloo Maple, the company that authored and sold the Maple software, for further details.” Pet. 16–17 (citing Ex. 1008, 44; Ex. 1006 ¶ 49).⁷ Petitioner explains that

Waterloo Maple published the Maple Guide, and a POSITA would have been motivated to read the Maple Guide to learn more about Maple. The teaching in the Distributed Maple Publications that Distributed Maple utilized Maple, including its

⁷ Describing Distributed Maple as using “a conventional Maple frontend,” Schreiner1 states that “‘Maple’ is a registered of ‘Waterloo Maple Inc.’” Ex. 1008, 7. Schreiner1 also cites <http://www.maplesoft.com> under a listing of reference sources, listing “Maple, W., Maple 6, 2001” as one such reference source. *Id.* at 44.

IPR2021-00108
Patent 8,676,877 B2

kernel and libraries, provides a POSITA with a strong, express motivation to combine the features described in the Distributed Maple Publications with the features of Maple, as described in the Maple Guide.

Id. at 17 (citing Ex. 1005 ¶ 47).

Petitioner also contends that “the references . . . were publicly available on the same webpage – <http://www.risc.unilinz.ac.at/software/distmaple>, which was cited by Schreiner1 and date-stamped and archived by the Internet Archive, and is submitted as Exhibits 1024 and 1025.” Pet. 16 (citing Ex. 1008, 5–6; Ex. 1009, 4 (Abstract); Ex. 1005 ¶¶ 45–46; Ex. 1006 ¶¶ 24–25; Ex. 1024; Ex. 1025). As noted above, Schreiner1 states that “[b]oth the Distributed Maple system itself and the library of parallel version of . . . Maple algorithms are in stable versions freely available under the GNU Library General Public License at <http://www.risc.uni-linz.ac.at/software/distmaple>.” Ex. 1008, 5. Depending on the specific alleged prior art reference, Petitioner provides other motivation as detailed further below.

Claim’s 1 preamble recites “[a] system for performing an instruction received from a front end by executing commands on one or more special purpose microprocessors, the system comprising.” *See* Pet. 22; *supra* § I.D. Petitioner contends that even if the preamble limits the claim, Schreiner1 “contains actual screenshots of a Maple user interface frontend, showing instructions entered by the user after each ‘>’ prompt.” *Id.* at 24 (reproducing Ex. 1008, 6–7 (screen shot of a user interface display); citing Ex. 1005 ¶ 64). Petitioner also annotates and reproduces Schreiner1’s Figure 1 (reproduced below and *see supra* § II.D.1), which shows a “user interface” connected to first node that includes a Maple kernel. *Id.* at 23.

IPR2021-00108
Patent 8,676,877 B2

Petitioner explains that Schreiner1’s “nodes execute commands in response to the user instructions.” Pet. 25. As an example, Petitioner explains that “Schreiner1 describes a user entering the Distributed Maple ‘dist[all]’ instruction into the root Maple kernel user interface: ‘dist[all](*command*) lets the Maple statement command be executed on every Maple kernel connected to the distributed session.’” *Id.* (quoting Ex. 1008, 9; citing Ex. 1009, 30) (emphasis omitted). Petitioner also cites the example of the “dist[start] instruction” as an “instruction entered into the user interface.” *Id.* (citing Ex. 1008, 8, 10–12).

Petitioner relies on PCMagazine1 and PC Magazine2 as suggesting the use of special purpose microprocessors to run Maple V such as the “80387 coprocessor” or other math coprocessor. Pet. 26–27 (citing Ex. 1029; Ex. 1030). Petitioner contends that PC Magazine2 recommends a “math coprocessor.” *Id.*; see Ex. 1030, 6 (“While Maple will use a math coprocessor, one is not essential, as it is for Mathematica.”); Ex. 1029, 7 (Maple V “[r]equires” an “80387 coprocessor”). According to Petitioner, “a POSITA would have been motivated to read other materials on Maple V, including PC Magazine1 and PC Magazine2, which confirm that use of a special purpose math coprocessor was a basic feature of Maple V.” Pet. 27 (citing Ex. 1005 ¶ 69). Petitioner adds that it would have been obvious to perform Maple math processes using a special purpose math coprocessor. See *id.* (citing Ex. 1005 ¶ 69).

Claim 1 also recites the following limitations:

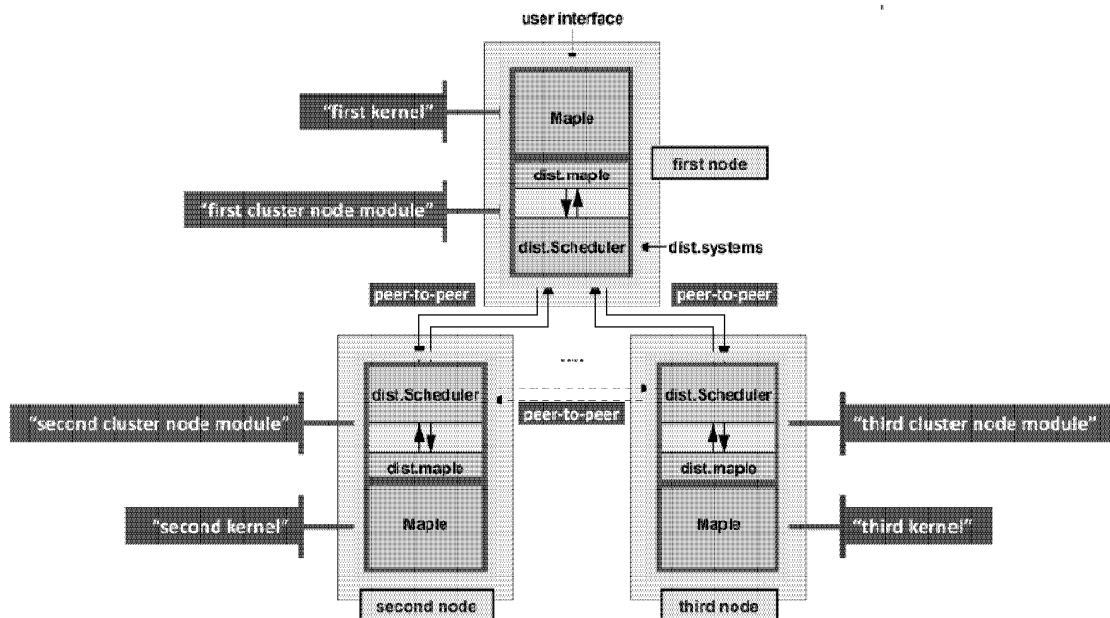
[b.] [i.] a plurality of nodes, wherein each node is configured to access a computer-readable memory system [ii.] comprising program code for a single-node kernel module, and wherein each single-node kernel module is configured to interpret instructions received by the single-node kernel module

IPR2021-00108
 Patent 8,676,877 B2

into commands that are executable by a special purpose microprocessor.

To address these limitations, Petitioner annotates Schreiner1's

Figure 1 as follows (Pet. 28):



According to Petitioner's annotations and descriptions, Figure 1 above shows single-node kernel modules (first kernel, second kernel, and third kernel) at each node (first node, second node, and third node), with "a processor executing a Maple kernel and Distributed Maple stored in memory." Pet. 29 (citing Ex. 1008, 9; Ex. 1005 ¶ 72). Petitioner explains that "[t]he Maple 'kernel consists of highly optimized C code.'" *Id.* (quoting Ex. 1011, 98; citing *id.* at 201 ("program such as Maple"), 277 ("either with Maple or with another program"), 11, 40, 102, 272 (Maple programs)). Petitioner explains that "Distributed Maple is also a software program . . . likewise installed in storage and loaded in memory for execution by one or more programs." *Id.* (citing Ex. 1005 ¶ 74; Ex. 1008, 5, 7–22 (describing

IPR2021-00108
 Patent 8,676,877 B2

the software system), 22–42 (describing application programs); Ex. 1009, 9; Ex. 1015; Ex. 1016).

Petitioner contends that “Schreiner1 teaches the use of Maple as a single-node kernel” and “[e]ach Maple kernel was configured to interpret user instructions.” Pet. 30 (citing Ex. 1008, Abstract, 3, 8–9, Fig. 1; Ex. 1005 ¶ 77). Petitioner quotes the Maple Guide, stating the Maple kernel “contains fundamental and primitive commands: *the Maple language interpreter (which converts the commands you type into machine instructions your computer processor can understand)*, algorithms for numerical calculation, and routines to display results and perform other input and output operations.” *Id.* at 31 (quoting Ex. 1011, 98; citing Ex. 1005 ¶ 79) (emphasis by Petitioner). Petitioner adds that Schreiner1 describes “a 128 processor SGI Origin 3800 distributed shared memory multiprocessor” for implementing Distributed Maple and Maple. *Id.* at 29 (citing Ex. 1008, 25).

Petitioner also relies on Schreiner2 as “teach[ing] that high-level commands are interpreted by the kernel into lower-level code for execution by the processors.” Pet. 31 (citing Ex. 1009, 7, 15, 30). According to Petitioner, “Schreiner1 . . . discloses using Distributed Maple with a Mathematica kernel, exactly like in the embodiments of the ’877 patent.” *Id.* at 32 (citing Ex. 1008, 4; Ex. 1005 ¶ 82).

Claim 1 also recites the following limitations:

[c.] [i.] a plurality of cluster node modules, wherein [ii.] each cluster node module is stored in a computer-readable memory system and [iii.] configured to communicate with a single-node kernel and with one or more other cluster node modules, [iv.] to accept instructions, and to interpret at least some of the

IPR2021-00108
 Patent 8,676,877 B2

instructions such that the plurality of cluster node modules communicate with one another in order to act as a cluster in executing commands using one or more hardware processors.

Generally addressing the cluster node modules and single-node kernel limitations, Petitioner annotates Schreiner1's Figure 1 (as reproduced above) to depict first, second, and third cluster node modules (blue) communicating in a peer-to-peer fashion with each other and communicating with single-node Maple kernels (red). *See* Pet. 28 (annotating Schreiner's Figure 1), 34 (same). Petitioner explains that Schreiner's "Figure 1 depicts the dist.maple and dist.Scheduler components of each cluster node module." *Id.* at 34 (citing Ex. 1008, Fig. 1; Ex. 1005 ¶ 86).

According to Petitioner, "[t]he dist.Scheduler and dist.maple modules work together [as a (blue) cluster node module] to provide communication capabilities: dist.Scheduler 'coordinates node interaction,' and dist.maple 'implements the interface between the [red] kernel [Maple] and the scheduler.'" Pet. 34 (quoting Ex. 1008, 8). To support its showing, Petitioner quotes the following passage from Schreiner1: "The Maple kernel is a single-threaded process which communicates by a simple communication protocol with the scheduler on the same node. All capabilities for parallel and distributed program execution are embedded in this scheduler." *Id.* (quoting Ex. 1008, 12).

Petitioner also quotes from Schreiner2 to describe the scheduler and Maple interface:

Scheduler A scheduler program manages the node interaction and schedules tasks among nodes. This program is implemented by a Java class library with main class dist.Scheduler and is independent of Maple. The initial scheduler process reads all application-specific information from the configuration file

IPR2021-00108
 Patent 8,676,877 B2

dist.systems and may then start instances of the scheduler on other machines with which it communicates via sockets.

Maple Interface The package dist.maple running on each Maple kernel implements the interface between Maple and the scheduler. Communication between both components is based on pipes (named pipes for the Maple kernel connected to the user interface and standard input/output streams for the backend kernels). During a session, additional socket connections between remote scheduler instances are dynamically established on demand.

Pet. 35 (quoting Ex. 1009, 24; citing Ex. 1005 ¶ 87).

Addressing limitation c.ii, Petitioner relies on its showing above with respect to limitation b.i, contending that the “[t]he Distributed Maple software was installed in a storage device and loaded into memory during operation.” Pet. 35.

Addressing limitation c.iii, Petitioner explains that the scheduler sends some of the Distributed Maple commands to all kernels. Pet. 36–41 (citing Ex. 1008, 9–10, 12, 14; Ex. 1009, 15). As an example, Petitioner quotes Schreiner¹ as follows: “All remote schedulers send new tasks to the root node scheduler which distributes them among all machines.” *Id.* at 37 (quoting Ex. 1008, 14). “[T]he scheduler accepts tasks from the Maple process and schedules these tasks among any node connected to the session.” *Id.* (quoting Ex. 1008, 14). Petitioner also contends that “[t]he dist.maple component ‘implements the interface between kernel and scheduler,’” providing the final link in sending commands to the kernels. *Id.* at 36 (quoting Ex. 1008, 8).

Relying on its annotated version of Schreiner’s Figure 1 (as reproduced above), Petitioner provides evidence that each cluster node module sends commands to other cluster node modules such that they

IPR2021-00108
 Patent 8,676,877 B2

communicate with each other in a peer-to-peer fashion. *See* Pet. 37–40 (citing Ex. 1008, 8–10, 14, Fig. 1 (arrows showing communications between dist.Scheduler), Fig. 5 (similar); Ex. 1010, 5, Fig. 5 (similar)).

As one example, Petitioner explains as follows:

Schreiner1 teaches that the scheduler components of the cluster node modules implement peer-to-peer communication comprising direct message passing between peer nodes: “[A]ll nodes know of each other, i.e. a node knows the address of a machine and the number of a port on which (a thread of) the remote scheduler is listening for connection requests. When a node needs to send a message to one of its peers, it can thus establish a direct connection for message transfers.”

Pet. 39 (quoting Ex. 1008, 13; citing Ex. 1008, 17; Ex. 1005 ¶ 97) (alteration by Petitioner).

As noted above, claim 1 also recites limitation c.iv, “a plurality of cluster node modules . . . configured . . . to accept instructions, and to interpret at least some of the instructions such that the plurality of cluster node modules communicate with one another in order to act as a cluster in executing commands using one or more hardware processors.” Claim 1 does not explicitly require instructions from a user interface. To the extent claim 1 requires a user interface (or front end) to send instructions, Petitioner relies on the user interface in Schreiner1’s Figure 1 (reproduced above) and provides other citations to Schreiner1.⁸ *See* Pet. 43 (citing Ex. 1008, 7 (“The user interacts with Distributed Maple via a conventional Maple frontend

⁸ In contrast to independent claim 1, independent claim 10 specifically recites “communicating an instruction from a front end to one or more cluster node modules” in the body of the claim. *See supra* §§ I.E; II.C). To address claim 10, Petitioner partly relies on its showing regarding the user interface in connection with claim 1 as discussed herein. *See* Pet. 64–65.

IPR2021-00108
 Patent 8,676,877 B2

(text or graphical), i.e. she operates within the familiar Maple environment for writing and executing parallel programs.”)).

Regarding the limitation of the cluster node modules “accept[ing] instructions, and to . . . communicate with one another in order to act as a cluster in executing commands,” Petitioner contends that “the root dist.maple component waits for Distributed Maple ‘dist’ message-passing commands from the user interface, and forwards them to other cluster node modules using its scheduler” and “the cluster node modules . . . act as a cluster” to execute the commands. Pet. 42–43 (citing Ex. 1008, 9–12).

Petitioner explains as follows:

Fig. 1 . . . shows a command from “user interface” to the root kernel. If the command is a Distributed Maple instruction (e.g., dist[all], dist[start], dist[wait], etc.), the dist.maple component accepts the instruction and passes it to the scheduler component, as shown by the arrow directed from dist.maple to the scheduler component of the root cluster node module.

Pet. 43 (citing Ex. 1008, Fig. 1; Ex. 1005 ¶ 102).

Relying further on Schreiner1, Petitioner provides the example of a “dist[all]” command sent from a user interface and interpreted by cluster node modules such that they act as a cluster:

The instructions from the user interface are interpreted by the cluster node modules “such that the plurality of cluster node modules communicate with one another in order to act as a cluster,” as the claim requires. For example, Schreiner1 describes a user entering the Distributed Maple “dist[all]” instruction into the user interface: “dist[all](*command*) lets the Maple statement *command* be executed on every Maple kernel connected to the distributed session.” EX-1008, [9]; EX-1009, 30. A POSITA understands that, in order for this Distributed Maple instruction and its embedded Maple statement (“*command*”) to be executed on every kernel, the instruction is sent to, and accepted by, every cluster node module, which then interprets it in order to execute

IPR2021-00108
 Patent 8,676,877 B2

it. Once each node evaluates the Maple statement, it communicates the result back to the node that issued the dist[all] instruction. Accordingly, the cluster node modules interpret the dist[all] command such that the nodes communicate to evaluate a Maple statement in parallel, thereby satisfying this claim element. EX-1005, ¶104.

Pet. 44.

Petitioner provides another example as disclosed in Schreiner1. *See* Pet. 45–46 (relying on a “dist[start] instruction” entered from a “user interface,” and “interpreted by the root dist.maple component as a *task*, which the root scheduler can pass to other cluster nodes for execution.” (citing Ex. 1008, 8, 10, 14; Ex. 1009, 30)).

Patent Owner responds by arguing that in Schreiner1, the root node is a master node and it schedules tasks between other nodes. *See* Prelim. Resp. 23–24. Accordingly, Patent Owner argues that Schreiner1 does not teach limitation c.iv under its proposed claim construction of “cluster node modules.” *See id.*; *supra* § II.C. For example, Patent Owner argues as follows: “[B]ecause the purported cluster node modules of the prior art do not ‘communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node,’ they do not satisfy the ‘cluster node module’ limitation as properly construed.” Prelim. Resp. 24. Contrary to this line of argument, as determined above for purposes of institution, the preliminary record does not support Patent Owner’s claim construction and does not require “communicating . . . without the tasks and data being required to go through a central server or master node.” *See supra* § II.C.

The final limitations of claim 1 follow:

[d.] “a communications system configured to connect the

IPR2021-00108
Patent 8,676,877 B2

plurality of nodes;”

[e.] “wherein the plurality of cluster node modules cooperate to interpret and translate, as needed, the instruction for execution by a plurality of single-node kernel modules, and

[f.] wherein at least one of the plurality of cluster node modules returns a result to the user interface.

For limitation d, Petitioner quotes Schreiner1’s disclosure of “networked environments” for the Distributed Maple software. *See* Pet. 47–48 (quoting Ex. 1008, 3). Petitioner also relies on other teachings in Schreiner1 describing computer nodes with processors connected in a network. *See id.* (citing Ex. 1008, 23, 25; Ex. 1005 ¶ 110).

For limitation e, Petitioner relies partly on its discussion of interpreting and translating Distributed Maple commands in connection with limitations b.ii, c.iii, and c.iv above. *See* Pet. 48–51. According to Petitioner, “the dist.maple and scheduler components interpret dist[all] and dist[start] commands for execution by the plurality of Maple kernels.” *Id.* at 48–49. Petitioner also relies on the following load balancing disclosure for “cooperation” as claimed:

Moreover, “[a]ll remote schedulers send new tasks to the root node scheduler which distributes them among all machines” using a load-balancing scheme. EX-1008, 316. Under this scheme, “a remote scheduler asks for new tasks whenever the number of received but not yet started tasks falls below a lower bound.” *Id.* This is yet another example of cooperation – in order for a task to be executed, the cluster node modules cooperate to establish a load balancing scheme to distribute and interpret or translate task commands for execution. EX-1005, ¶116.

Id. at 51 (alteration by Petitioner).

For limitation f, Petitioner reproduces “a screenshot of a session in which the user inputs several instructions (highlighted in green) and the

IPR2021-00108
Patent 8,676,877 B2

cluster node modules then return the result to the user interface (highlighted in orange).” *See* Pet. 52 (citing Ex. 1008, 8; Ex. 1009, 5; Ex. 1005 ¶¶ 120–121). Petitioner also relies on a graphical output example for display in Schreiner1’s user interface. *See id.* at 52–53 (reproducing Ex. 1008, Fig. 13).

Building on its showing with respect to claim 1, Petitioner’s showing with respect to independent claims 8 and 10 somewhat tracks its showing with respect to independent claim 1. Pet. 53–62, 64–68. Petitioner supports its showing by citations to the record evidence, including the testimony of Dr. Tufo. *See id.* The same remarks apply to dependent claims 9 and 11. *See id.* at 62–64, 68–70.

Patent Owner does not address Petitioner’s showing with respect to claims 8 and 9. Patent Owner addresses independent claim 10 and its dependent claim 11 by asserting that claim 10 precludes passing an instruction indirectly from a user interface, via a kernel, to a cluster node module. *See* Prelim. Resp. 25–29. Specifically, Patent Owner argues that “[t]he Petition does not show that the prior art discloses the required relative order of processing instructions” under its proposed claim construction. *Id.* at 25 (Limitations in claim 10 “should be construed to require the following relative order . . . front end → cluster node module → kernel.”) As set forth above, however, the preliminary record does not support Patent Owner’s narrow claim construction that precludes an intervening kernel between the front end and cluster node modules. *See supra* § II.C.

Based on the foregoing discussion, after considering other arguments and evidence presented by Patent Owner as addressed further below, the

IPR2021-00108
 Patent 8,676,877 B2

Petition presents a sufficient showing supported by the record with respect to claims 1 and 8–11.

b. Public Accessibility of the Distributed Maple Code

Patent Owner also generally argues, with respect to all of the claim limitations, that Petitioner failed to show that the Distributed Maple Code references were publically accessible prior to the date of the invention. *See* Prelim. Resp. 20–23 (noting that Distributed Maple Code includes Exhibits 1012–1018). However, Petitioner relies on the Distributed Maple Code references, which describe the source code for the Distributed Maple Code referenced in Schreiner1, Schreiner2, Schreiner3, the Maple Guide, and other references, to *support* its showing based on the latter references. *See, e.g.*, Pet. 31 (“This is evident in the dist.maple code as well.”).

Patent Owner does not directly argue that the other references do not support institution sufficiently without the Distributed Maple Code (i.e., source code). *See* Prelim. Resp. 21 (“The Board *may* deny the Petition on that basis alone.”) (emphasis added). In any event, addressing the Distributed Maple Code, Patent Owner contends that Petitioner relies on the uncorroborated testimony of Dr. Schreiner that he posted the Distributed Maple Code “in 2003 on the public website of Research Institute for Symbolic Computation (‘RISC’), where the references allegedly could be downloaded through the webpage shown in Exhibit 1024.” *Id.* at 21 (arguing “corroboration is required of a witness’s testimony about his own allegedly invalidating activities” (citing *Finnigan Corp. v. ITC*, 180 F.3d 1354, 1366 (Fed. Cir. 1999))). However, Petitioner also cites Schreiner1 as referring to the source code and listing the noted website, and Patent Owner acknowledges that Schreiner1 lists the website. *See* Pet. 15 (citing

IPR2021-00108
 Patent 8,676,877 B2

Ex. 1008, 5; Ex. 1006 ¶¶ 22–23; Ex. 1005 ¶ 45); Prelim. Resp. 21–22; Ex. 1008, 5–6. Nevertheless, Patent Owner contends that “[t]he most the evidence submitted by Petitioner shows is that [Dr.] Schreiner himself, or possibly others who helped create the Distributed Maple Code or already knew of its existence, may have been able to locate whatever version was posted at that time.” Prelim. Resp. 23.

This line of argument downplays that Schreiner¹, published in the Journal of Symbolic Computation in 2003, would have pointed interested artisans to the website listed therein in order to obtain the “Distributed Maple system itself,” the main subject of Schreiner¹ and described as “freely available.” See Ex. 1008, 5. Also, the Internet Archive screenshot, Exhibit 1024, describes “Distributed Maple” and lists the same website as published in Schreiner¹, and describes the website as “[m]aintained by: Wolfgang Schreiner, Last Modification: July 14, 2003.” See also Ex. 1006 ¶ 24 (Dr. Schreiner noting that the Internet Archive screenshot states “Last Modification: July 14, 2003” and testifying “[t]hat is consistent with my recollection of the time when I last modified this page” (citing Ex. 1024)); Ex. 1025 (similar Internet Archive screenshot evidence). This evidence corroborates Dr. Schreiner’s testimony as to the timeframe he uploaded the source code to the RISC website.

Dr. Schreiner also testifies that “[i]t has been my practice to check, from time to time, whether my software was accessible through Google search results, and I did this prior to 2005 for these particular web pages and confirm that my Distributed Maple papers and software were accessible through Google searches.” Ex. 1006 ¶ 21. Patent Owner argues that “Petitioner relies entirely upon Schreiner’s memory from more than *seven*

IPR2021-00108
 Patent 8,676,877 B2

years ago to suggest the version of Distributed Maple Code filed in this IPR was publicly accessible back then.”⁹ Prelim. Resp. 21 (emphasis added). However, as indicated above, Schreiner1, coauthored by Dr. Schreiner with others, and the Internet Archive documents, corroborate Dr. Schreiner’s testimony about uploading the software on the RISC website.¹⁰ During trial, Patent Owner will have the opportunity to cross-examine Dr. Schreiner, including regarding Google searches and his memory, assuming for the sake of argument that the ability to search the RISC website using Google is relevant to show public accessibility of the source code.

On this preliminary record, sufficient evidence exists to show that the Distributed Maple Code was publically available in 2003 and thereafter up to the date of the invention in 2006. *See* Ex. 1001, code (60) (listing the filing date of a provisional application as October 11, 2006). Patent Owner does not dispute that some form of the source code existed prior to the date of the invention. To the extent the source code may have changed over the relevant timeframe as Patent Owner argues (*see* Prelim. Resp. 21–22), the parties will have the opportunity to address the materiality of any such changes as they relate to specific claim limitations or evidence during trial.

Even if the source code was not publicly available at the relevant time, Petitioner’s reliance on it as extrinsic evidence solely to support its showing of how the Distributed Maple system operated at the time of the invention would be proper. *See In re Baxter Travenol Labs.*, 952 F.2d 388,

⁹ Given the timeframe of 2003 to 2005 at issue here, Patent Owner probably intends “seventeen years ago” instead of “seven years ago.”

¹⁰ As noted above (§ II.D.1), two others co-authored Schreiner1 with Dr. Schreiner.

IPR2021-00108
Patent 8,676,877 B2

390 (Fed. Cir. 1991) (extrinsic evidence may be used to explain what a reference discloses); *Hospira v. Fresenius Kabi USA*, 946 F.3d 1322, 1329 (Fed. Cir. 2020) (“Extrinsic evidence can be used to demonstrate what is ‘necessarily present’ in a prior art embodiment even if the extrinsic evidence is not itself prior art.”). In any event, on this preliminary record for purposes of institution, after considering Petitioner’s showing and Patent Owner’s arguments and objective evidence of nonobviousness (as discussed further below), Petitioner sufficiently shows that Schreiner1, Schreiner2, Schreiner3, the Maple Guide, PC Magazine1, and PC Magazine 2, teach claim 1 with or without the supporting source code as disclosed in the Distributed Maple Code.¹¹

c. Alleged Objective Evidence of Nonobviousness

Objective evidence of nonobviousness “may often be the most probative and cogent evidence in the record” and “may often establish that an invention appearing to have been obvious in light of the prior art was not.” *Transocean Offshore Deepwater Drilling, Inc. v. Maersk Drilling*

¹¹ Patent Owner also contends that Petitioner’s declarant impermissibly relies on “public use” information—i.e., information about commercial embodiments. *See* Prelim. Resp. 49–53; PO Sur-reply 10. Petitioner disagrees. Pet. Reply 9–10. For purposes of institution on this preliminary record, Petitioner sufficiently shows that the cited Exhibits support Dr. Tufo’s testimony and at least teach claim 1 without improper reliance on public use information. *See id.* (mapping support for Dr. Tufo’s testimony at Ex. 1005 ¶¶ 37–40, 70–76 to Ex. 1008–Ex. 1011). (To provide further support for its position, Petitioner also cites an alleged Exhibit supplied by Patent Owner (i.e., “EX-2005, 2”). *See* Pet. Reply 10. However, this Exhibit does not appear in the record so any argument premised on it provides no added support for Petitioner.)

IPR2021-00108
Patent 8,676,877 B2

USA, Inc., 699 F.3d 1340, 1349 (Fed. Cir. 2012) (citing *Stratoflex, Inc. v. Aeroquip Corp.*, 713 F.2d 1530, 1538 (Fed. Cir. 1983)).

Patent Owner argues that objective evidence regarding its SEM and SET products supports the nonobviousness of the challenged claims. Prelim. Resp. 33–46. Patent Owner alleges evidence of long-felt and unresolved need, failure by others, praise by others, skepticism of others, and copying. *Id.*

(i.) *Nexus*

Nexus is a legally and factually sufficient connection between the objective evidence and the claimed invention requiring a tribunal to consider the objective evidence in determining nonobviousness. *Demaco Corp. v. F. Von Langsdorff Licensing Ltd.*, 851 F.2d 1387, 1392 (Fed. Cir. 1988) (“Once a prima facie case of nexus is made the court must consider the evidence adduced on both sides of the question, with such weight as is warranted.”). “The patentee bears the burden of showing that a nexus exists” *WMS Gaming Inc. v. Int’l Game Tech.*, 184 F.3d 1339, 1359 (Fed. Cir. 1999).

Patent Owner contends that “traditional parallel-computing architectures [were] notoriously difficult, time-consuming, and expensive.” Prelim. Resp. 32 (citing Ex. 2036 ¶¶ 13–18; Ex. 2007 ¶ 8; Ex. 2008 ¶ 21). Patent Owner contends that programmers typically generated parallel code by breaking up and converting serial code “for execution on each of the nodes of a parallel computer.” *Id.* at 33. According to Patent Owner, “there was a long-felt but unmet need for a way to unlock the performance advantages of cluster computing without requiring specialized expertise or excessive time, effort, and cost (i.e., a need to break the

IPR2021-00108
 Patent 8,676,877 B2

performance/programming barrier).” *Id.* at 34 (citing Ex. 2036 ¶¶ 10–19; Ex. 2007 ¶¶ 8, 11; Ex. 2008 ¶¶ 16–23).

Patent Owner alleges it developed “a cluster-computing architecture, used in its SEMTM and SETTM’s products, that met this long-felt, unmet need.” Prelim. Resp. 34–35. Patent Owner asserts that SEMTM and SETTM “enable[] parallel execution of Mathematica and more general applications, respectively . . . without extensive specialized programming expertise, or the excessive investment of time and effort demanded by traditional parallel computing architectures.” *Id.* at 35–36 (citing Ex. 2036 ¶¶ 20–30; Ex. 2007 ¶¶ 10, 12 (discussing SEMTM); Ex. 2008 ¶¶ 25–28 (discussing SETTM)).

As further described by Patent Owner, to meet this “long-felt, unmet need,” “[t]he new architecture interposed a communication layer between the front end user interface and the kernels running on each node of the cluster, or back end.” Prelim. Resp. 34–35. Patent Owner contends that “[t]he unique SEMTM and SETTM cluster-computing architecture embodied by the challenged claims is the reason that SEMTM and SETTM were able to meet the long-felt but previously unmet need.” *Id.* at 36 (citing Ex. 2036 ¶¶ 24, 27–29; Ex. 2001 ¶¶ 65–70). Patent Owner argues this alleged claimed architecture obtained “superior performance” and “ease of use” that “were unexpected” and “surprising.” *Id.* at 36–37. Similarly, Patent Owner argues that “SEMTM and SETTM succeeded where others failed” and received praise, and “many were skeptical that the SEMTM architecture could truly enable the high performance of cluster computing without requiring specialized expertise or excessive time and effort.” *Id.* at 38–39.

Patent Owner argues that “[t]here is sufficient nexus between the objective evidence related to SEMTM and SETTM and the challenged claims”

IPR2021-00108
 Patent 8,676,877 B2

because the “SEM™ and SET™ products practice at least the challenged independent claims.” Prelim. Resp. 40. Patent Owner argues that “[e]ach of the objective indicia of non-obviousness results from the SEM™ and SET™ architecture embodied by the challenged claims.” *Id.* at 44. Patent Owner argues that “[o]ther parallel-computing architectures failed to meet the long-felt, unmet need precisely because they lacked SEM™ and SET™’s claimed architecture.” *Id.* (citing Ex. 2036 ¶ 37). Patent Owner argues the “claimed architecture” is responsible for all of its asserted “objective indicia of nonobviousness.” *Id.* (citing Ex. 2036 ¶¶ 21–31, 39–42, 46, 47).

Nevertheless, even if the SEM™ and SET™ products fall within the broad scope of the challenged claims, for similar reasons to those underlying the claim construction as set forth above (§ II.C), the challenged claims do not require the “unique . . . architecture” of SEM™ and SET™. *See* Prelim. Resp. 36. In simple terms, the challenged claims are not architecture-specific. *See MeadWestVaco Corp. v. Rexam Beauty and Closures, Inc.*, 731 F.3d 1258, 1264 (Fed. Circ. 2013) (error to consider “*secondary considerations of non-obvious [that] involved only fragrance-specific uses*,” when “*claims now at issue are not fragrance-specific*” (emphasis added)). The court in *MeadWestVaco* held that the district court erred because it “credited evidence advanced to show long-felt need and commercial success specific to the perfume industry,” and the claims were not limited to fragrance-specific dispensers. *See id.* (reasoning that ““objective evidence of non-obviousness must be commensurate in scope with the claims which the evidence is offered to support””) (quoting *Asyst Techs., Inc. v. Emtrak, Inc.*, 544 F.3d 1310, 1316 (Fed. Cir. 2008) (internal quote citation omitted)); *see also Brown & Williamson Tobacco Corp. v. Philip Morris Inc.*, 229 F.3d

IPR2021-00108
 Patent 8,676,877 B2

1120, 1130 (Fed. Cir. 2000) (stating the presumption that commercial success is due to the patented invention applies “if the marketed product embodies the claimed features, and is coextensive with them”).

The Federal Circuit recently addressed nexus in two related Board cases, *Fox Factory, Inc. v. SRAM, LLC*, 944 F.3d 1366, 1373–78 (Fed. Cir. 2019) (“*Fox Factory I*”) and *Fox Factory, Inc. v. SRAM, LLC*, 813 F. App’x 539, 542 (Fed. Cir. 2020) (*Fox Factory II*). In *Fox Factory II*, the court characterized the Board’s holding underlying *Fox Factory I*, as follows:

In [*Fox Factory I*], this court held that the Board misapplied the legal requirement, incumbent upon patent owners, of showing a nexus between evidence of secondary considerations and the obviousness of the claims of that patent—in particular, the requirement that the product from which the secondary considerations arose is “coextensive” with the claimed invention. *Fox Factory*, 944 F.3d at 1373–78. Contrary to the Board’s view, we reaffirmed in that case that a product is not coextensive with a claimed invention simply because it falls within the scope of the claim.

Fox Factory II, 813 F. App’x at 542. In other words, no presumption of nexus exists for products that simply fall in the broad scope of the claims if the products are not coextensive with the claims.

As noted above, according to Patent Owner, “the new architecture interposed a communication layer between the front end user interface and the kernels running on each node of the cluster, or back end”—i.e., a “unique . . . architecture.” *See* Prelim. Resp. 34–35, 37. As construed above, however, the challenged claims do not require a communication layer between the user interface and kernels—the central feature relied upon by Patent Owner for its objective evidence of nonobviousness. *See supra* § II.C; Pet. Reply 6 (arguing that Dr. Dager’s testimony “rests on an

IPR2021-00108
Patent 8,676,877 B2

incorrect claim construction”). Therefore, on this preliminary record, because the claims do not require this allegedly “unique . . . architecture,” the claims are not coextensive (or reasonably commensurate in scope) with the products and other asserted evidence, so no nexus exists.

For the foregoing reasons and on this preliminary record, Patent Owner fails to meet its burden of establishing a nexus between the objective evidence regarding its SEM and SET products and the challenged claims of the ’877 patent. We, therefore, do not accord substantial weight to such evidence. *Fox Factory I*, 944 F.3d at 1373. For the sake of completeness, however, we address Patent Owner’s remaining allegations relating to objective indicia of nonobviousness.

(ii) Long-Felt Need and Failure of Others

“The existence of a long-felt but unsolved need that is met by the claimed invention is . . . objective evidence of non-obviousness.”

Millennium Pharms., Inc. v. Sandoz Inc., 862 F.3d 1356, 1369 (Fed. Cir. 2017) (citing *In re Cyclobenzaprine Hydrochloride Extended-Release Capsule Patent Litig.*, 676 F.3d 1063, 1081–83 (Fed. Cir. 2012)).

“Long[-]felt need is closely related to the failure of others. Evidence is particularly probative of obviousness when it demonstrates both that a demand existed for the patented invention, and that others tried but failed to satisfy that demand.” *Cyclobenzaprine*, 676 F.3d at 1082.

Patent Owner argues “there was a long-felt but unmet need for a way to unlock the performance advantages of cluster computing without requiring specialized expertise or excessive time, effort, and cost.” Prelim. Resp. 34. Patent Owner argues that its SEM and SET products met this

IPR2021-00108
 Patent 8,676,877 B2

need. *Id.* at 35–37 (citing Ex. 2001 ¶¶ 65–70; Ex. 2007 ¶¶ 10, 12; Ex. 2008 ¶¶ 25–28; Ex. 2036 ¶¶ 10–30).

Petitioner argues that “[Dr.] Dauger provides no evidence or explanation for why interposing the communications software between the user interface and the kernel, as compared to connecting the communications software in some other way, would make any difference at all to the user.” Pet. Reply 6. Petitioner argues that this testimony is also unpersuasive because it relies on an incorrect claim interpretation that requires the cluster node modules to accept instructions from the user interface without the instructions first passing through any kernel. *Id.* at 6–8.

Patent Owner argues that Petitioner’s arguments regarding claim construction are outside the scope of our Order authorizing Petitioner to file its Reply. PO Sur-reply 6–9. To the contrary, as discussed in considering nexus above (§ II.D.6.c.ii), the scope of the claims is relevant to objective indicia of nonobviousness.

Regarding the failure of others, Patent Owner acknowledges that others developed automatic parallelizers and universal compilers that converted serial code to parallel code, but argues that none achieved performance comparable to traditional parallel-computing architectures. Prelim. Resp. 32; *see also* PO Sur-reply 3.

Petitioner argues that “P[atent] O[wner]’s argument is irrelevant because the claims neither recite ‘automatic’ or ‘universal’ parallelization nor require a specific level of optimization or advantageousness.” Pet. Reply 2 (citing *ABT Sys., LLC v. Emerson Elec. Co.*, 797 F.3d 1350, 1362 (Fed. Cir. 2015)).

IPR2021-00108
 Patent 8,676,877 B2

Patent Owner relies on the testimony of Dr. Dauger, Dr. Bhansali, and Mr. Bancroft to support its arguments about a long-felt, unmet need. *See* Prelim. Resp. 32–36. However, none of these declarants establishes that others unsuccessfully attempted to solve the problem. Dr. Dauger states that “[n]umerous others had tried to implement automatic parallelizers or universal compilers that would take serial object code as input and output object parallel code,” but “[n]one of these efforts succeeded to produce accurate parallel code that was sufficiently optimized or advantageous enough to catch on.” Ex. 2036 ¶ 37. These conclusory and uncorroborated statements fail to establish that others actually tried to solve the asserted problem. The testimony of the other declarants fares no better. Dr. Bhansali merely states that he was not aware of any other products like the SEM product. Ex. 2007 ¶ 11. Mr. Bancroft states that he “had heard rumo[rs] of people trying to develop a universal parallelizer that could be used to automatically parallelize serial code.” Ex. 2008 ¶ 30. None of this testimony persuasively establishes that others actually tried and failed to solve the problem asserted by Patent Owner.

Accordingly, for the foregoing reasons and on this preliminary record, Patent Owner’s evidence of long-felt need and failure of others is weak.

(iii) Unexpected Results

“If a patent challenger makes a prima facie showing of obviousness, the owner may rebut based on ‘unexpected results’ by demonstrating ‘that the claimed invention exhibits some superior property or advantage that a person of ordinary skill in the relevant art would have found surprising or unexpected.’” *Procter & Gamble Co. v. Teva Pharms. USA, Inc.*, 566 F.3d 989, 994 (Fed. Cir. 2009) (quoting *In re Soni*, 54 F.3d 746, 750 (Fed. Cir.

IPR2021-00108
 Patent 8,676,877 B2

1995)). “To be particularly probative, evidence of unexpected results must establish that there is a difference between the results obtained and those of the closest prior art, and that the difference would not have been expected by one of ordinary skill in the art at the time of the invention.” *Bristol-Myers Squibb Co. v. Teva Pharms. USA, Inc.*, 752 F.3d 967, 977 (Fed. Cir. 2014); *see also Kao Corp. v. Unilever U.S., Inc.*, 441 F.3d 963, 970 (Fed. Cir. 2006) (“[W]hen unexpected results are used as evidence of nonobviousness, the results must be shown to be unexpected compared with the closest prior art.”) (quoting *In re Baxter Travenol Labs.*, 952 F.2d at 392).

Patent Owner argues that the performance and ease of use of its SEM and SET products was unexpected. Prelim. Resp. 36–38 (citing Ex. 2036 ¶¶ 38–41). Patent Owner argues that its SEM product outperformed gridMathematica. *Id.* at 37–38. Patent Owner argues that it was able to parallelize Wolfram Research’s Mathematica, Apple’s HD QuickTime Exporter, and Equalis’s Scilab using its SET product in a much shorter time period than expected. *Id.* at 38.

Petitioner argues that Patent Owner’s reliance on the testimony of Dr. Dauger is unpersuasive because Dr. Dauger is a listed inventor of the ’877 patent. Pet. Reply 1. Petitioner also argues that Dr. Dauger’s “testimony is also irrelevant because it compares the claimed invention against gridMathematica, not the ‘closest prior art.’” *Id.* (citing *In re Harris*, 409 F.3d 1339, 1344 (Fed. Cir. 2005); *Trs. of Columbia Univ. v. Illumina, Inc.*, 620 F. App’x 916, 922 (Fed. Cir. 2015)).

Patent Owner replies that “Dr. Dauger’s testimony was submitted under oath and penalty of perjury” and the other evidence cited in its Preliminary Response support its contention that the SEM and SET products

IPR2021-00108
 Patent 8,676,877 B2

exhibited surprising results. PO Sur-reply 1–2. Patent Owner also argues that it “chose the closest prior art by comparing SEM™ with gridMathematica and SET™ with the conventional method of parallelizing applications.” *Id.* at 2 (citing Ex. 2036 ¶ 40).

Patent Owner relies almost exclusively on the testimony of Dr. Dauger in asserting the surprising results of the SEM and SET products. *See* Prelim. Resp. 36–38 (citing Ex. 2036 ¶¶ 38–41). Dr. Dauger testifies the he was surprised that the SEM product performed better than gridMathematica and that Patent Owner was able to parallelize Mathematica “in one man-month.” Ex. 2036 ¶¶ 38–41. Dr. Dauger is an inventor of the ’877 patent, was Patent Owner’s CTO, and is currently a consultant employed by Patent Owner. Ex. 1001, code (75); Ex. 2036 ¶ 1; Ex. 2015, 2 (SET presentation I); Ex. 2018, 2 (SET presentation II). On this preliminary record, Dr. Dauger’s testimony about his personal surprise at the SEM and SET products that he helped create is unpersuasive to establish unexpected results of these products. *See In re Cree*, 818 F.3d 694, 702 (Fed. Cir. 2016) (citing *Power-One v. Artesyn Techs., Inc.*, 599 F.3d 1343, 1352 (Fed. Cir. 2010) (concluding that “self-serving statements from researchers about their own work” do not have the same credibility as statements made by disinterested parties). Notably, Patent Owner provides no evidence to corroborate Dr. Dauger’s assertions of unexpected results.

Additionally, Patent Owner provides no comparative testing against any prior art configuration, be it the closest or otherwise. Although Dr. Dauger testifies that some testing was performed (*see* Ex. 2036 ¶ 39), no documentation or data are provided from that testing to substantiate his assertions. This is the type of conclusory evidence that has been found

IPR2021-00108
Patent 8,676,877 B2

insufficient. *See, e.g., In re Lindner*, 457 F.2d 506, 508 (CCPA 1972) (“This court has said previously that mere lawyers’ arguments unsupported by factual evidence are insufficient to establish unexpected results. . . . Likewise, mere conclusory statements in the specification and affidavits are entitled to little weight when the Patent Office questions the efficacy of those statements.”).

Furthermore, Patent Owner does not persuasively argue that gridMathematica is the closest prior art. As noted by Petitioner, Patent Owner does not compare its products to the asserted references or the Distributed Maple system disclosed therein.

Accordingly, for the foregoing reasons and on this preliminary record, Patent Owner’s evidence of unexpected results is weak.

(iv) Industry Praise

“Evidence that the industry praised a claimed invention or a product that embodies the patent claims weighs against an assertion that the same claimed invention would have been obvious. Industry participants, especially competitors, are not likely to praise an obvious advance over the known art.” *Apple Inc. v. Samsung Elecs. Co.*, 839 F.3d 1034, 1053 (Fed. Cir. 2016) (en banc).

Relying on the asserted statements of Dr. Bhansali and Yuko Matsuda, Patent Owner argues that industry praise supports patentability of the ’877 patent claims. Prelim. Resp. 38–39. According to Patent Owner, Dr. Bhansali found the SEM product to be efficient for load balancing issues and Mr. Matsuda endorsed the SEM product. *Id.*

Petitioner argues that “[Dr.] Bhansali focuses on ‘load balancing,’” which “has no nexus because the patents do not assert that load balancing

IPR2021-00108
Patent 8,676,877 B2

was novel or non-obvious.” Pet. Reply 3–4 (citing *Kennametal, Inc. v. Ingersoll Cutting Tool Co.*, 780 F.3d 1376, 1385 (Fed. Cir. 2015)). Petitioner argues that Patent Owner’s proposed construction of “cluster node module” excludes the only method of load balancing disclosed in the ’877 patent. *Id.* at 4–5. Petitioner also argues that the references asserted in the Petition teach load balancing. *Id.* at 4.

Patent Owner argues that Petitioner’s arguments regarding claim construction are outside the scope of our Order authorizing Petitioner to file its Reply. PO Sur-reply 1, 4–6. To the contrary, as discussed in considering nexus above (§ II.D.6.c.ii), the scope of the claims is relevant to objective indicia of nonobviousness.

In order for evidence of industry praise to be probative of nonobviousness, the evidence must relate specifically to features of the claimed invention. *See Apple*, 839 F.3d at 1053–55 (discussing “substantial evidence of praise in the industry that specifically related to features of the claimed invention”). As argued by Patent Owner, Dr. Bhansali testifies that he found the SEM product to be efficient for load balancing issues, by which he means “*the distribution of different parts of an algorithm or application across different nodes* and the overall process of parallelizing the algorithm or application.” Ex. 2007 ¶ 12 (emphasis added). The ’877 patent refers to load balancing in a similar manner. *See* Ex. 1001, 21:8–55. As correctly noted by Petitioner, the challenged claims do not recite load balancing or otherwise require distributing commands among the nodes in a particular manner. Therefore, on this preliminary record, Dr. Bhansali’s testimony appears to focus on unclaimed features such that it is not probative of nonobviousness.

IPR2021-00108
 Patent 8,676,877 B2

Regarding the asserted statements made by Mr. Matsuda, Patent Owner cites to the Dauger Declaration rather than any submission endorsed by Mr. Matsuda. Prelim. Resp. 39 (citing Ex. 2036 ¶¶ 44–45). Dr. Dauber cites to a slide deck, which he appears to have prepared and the substance of which consists only of two quotations. Ex. 2036 ¶ 44 (citing Ex. 2018, 4); *see also* Ex. 2018, 2 (listing Dean E. Dauger, Ph.D. as the author). On this preliminary record, the uncorroborated statements of Dr. Dauber are unpersuasive to support the asserted statement of Mr. Matsuda.

Dr. Dauger also cites to Exhibit 2023, referring to it as a “white paper” written by Mr. Matsuda. Ex. 2036 ¶ 45 (citing Ex. 2023, 2). Initially, it is not clear what significance a “white paper” carries. Moreover, in the sentence cited by Patent Owner, Mr. Matsuda merely states that the SEM product “stands in an advantageous position” compared with an undefined “Parallel Computing Toolkit” when used with Mathematica. Ex. 2023, 2. This is not the type of competitor praise that courts have found to be indicative of non-obviousness. *See, e.g., Apple*, 839 F.3d at 1053–54 (discussing “numerous internal Samsung documents that both praised Apple’s slide to unlock feature and indicated that Samsung should modify its own phones to incorporate Apple’s slide to unlock feature”).

Accordingly, for the foregoing reasons and on this preliminary record, Patent Owner’s evidence of industry praise is weak.

(v) Skepticism

Evidence of industry skepticism weighs in favor of nonobviousness. *See United States v. Adams*, 383 U.S. 39, 52 (1966). “If industry participants or skilled artisans are skeptical about whether or how a problem could be solved or the workability of the claimed solution, it favors

IPR2021-00108
Patent 8,676,877 B2

nonobviousness.” *WBIP, LLC v. Kohler Co.*, 829 F.3d 1317, 1335 (Fed. Cir. 2016).

Patent Owner argues that experts expressed skepticism that the SEM and SET products would work. Prelim. Resp. 39–40. Regarding the SEM product, Patent Owner relies solely on the testimony of Dr. Bhansali and Mr. Bancroft. *Id.* at 39 (citing Ex. 2007 ¶ 8; Ex. 2008 ¶ 32). Regarding the SET product, Patent Owner relies on the opinion of one unidentified Department of Energy (“DOE”) “reviewer.” *Id.* at 40 (citing Ex. 2036 ¶ 52); *see also* PO Sur-reply 3–4.

Petitioner argues that rather than expressing skepticism that Patent Owner’s products would work, the DOE reviewers quoted in Patent Owner’s exhibits “expressed skepticism over the bold performance claims made by P[atent] O[wner].” Pet. Reply 2–3.

Regarding the SEM product, Patent Owner relies solely on statements of Dr. Bhansali and Mr. Bancroft regarding their personal experience with the SEM product. Prelim. Resp. 39 (citing Ex. 2007 ¶ 8; Ex. 2008 ¶ 32). Dr. Bhansali states that “[w]hen I first learned about SEM, I was uncertain whether the product would perform as promised.” Ex. 2007 ¶ 8. Dr. Bhansali states that he “graduated from Cal. Tech. with a dual B.S.-M.S. in physics, and engineering and applied science” and “received [a] Ph.D. in theoretical physics from Harvard University.” *Id.* ¶¶ 3–4. Neither Patent Owner nor Dr. Bhansali provide any detail about his industrial experience. Notably, Patent Owner did not file a CV for Dr. Bhansali in the record. Therefore, Patent Owner fails to establish that Dr. Bhansali is an artisan of ordinary skill with respect to parallel or distributed computing, and his opinion testimony carries little weight.

IPR2021-00108
 Patent 8,676,877 B2

Although Mr. Bancroft states that he provided the SEM product to “many experienced parallel programmers” (Ex. 2008 ¶ 32), he provides no information about these allegedly experienced programmers. Additionally, neither Patent Owner nor Mr. Bancroft provide his CV, so it is difficult to assess his credibility to provide technical testimony. *See* Ex. 2008.

Mr. Bancroft’s experience appears to involve business development matters rather than technical engineering or computer science research and development. *See id.* ¶¶ 4–14. Moreover, Mr. Bancroft is on Patent Owner’s Business Advisory Board. *Id.* ¶ 33. Therefore, it appears that Mr. Bancroft is not a disinterested party and may have economic or other interest in Patent Owner’s success in this proceeding. Accordingly, on this preliminary record, Mr. Bancroft’s opinion testimony carries little weight.

Regarding the SET product, Petitioner sufficiently shows that the DOE reviewers appear to indicate that the submissions they reviewed lacked sufficient detail for them to evaluate the performance assertions made in the submissions. *See* Pet. Reply 2–3. Patent Owner relies on the comments of “Reviewer 2” of Exhibit 2019. Prelim. Resp. 40 (citing Ex. 2036 ¶ 52; Ex. 2019, 2). This reviewer states that “[t]he proposal . . . provides no quantitative or qualitative evidence of (efficient or not) use of compute[r] resou[r]ces by SET.” Ex. 2019, 2. This reviewer also states that “[t]he applicants have not demonstrated quantitatively that their technology provides real results. . . . SET may prove to be the great success the applicants suggest, but there is no proof that it works on real CAD/CAM/CAE applications.” *Id.* at 3. Continuing, this reviewer states that “[t]he applicants haven’t clearly defined the nature of the plasma code, nor the effort in porting it to SET.” *Id.* Thus, the statements of Reviewer 2

IPR2021-00108
 Patent 8,676,877 B2

appear to stem from a lack of detail to assess the credibility of the assertions in the submissions.¹²

Other reviewers similarly identify a lack of detail in the submissions. *See* Ex. 2019, 4 (“[T]he main concepts of this proposal have not been presented in any substantial detail.” “There is no sound plan to showing that this SET-based approach can be commercially viable.”); Ex. 2021, 1 (“[T]here is no plan to compare the performance of the applications compared to their theoretical performance.”), 2 (“The auto parallelization tools have not provided high performance as they usually have too many generalizations to take advantage of a particular computing architecture. I do not have evidence that the SET tool is any different.”), 4 (“The applicant has provided a general outline of the comparison test approach, but further details in the work plan are needed.”; “While there is an overall projection of technical relevance, the lack of specificity in the proposed test situation presents a high level of uncertainty in achieving the more ambitious goals of this proposal.”); Ex. 2022, 3 (“The performance of SET in Linux and Mac OS operating environments, the type of efficiency increases achieved, and the strength and limitations of the SET approach are not adequately described.”), 4 (“The applicant has provided a general description of the technical problem and work plan, but specific details of the technical challenges to be encountered with the SET technology should be described in greater detail.”). Thus, to the extent one reviewer expressed skepticism that the SET product would perform as claimed, this evidence is undercut by

¹² The submissions to the DOE are not of record in this case.

IPR2021-00108
 Patent 8,676,877 B2

the overwhelming expression of a lack of detail provided in the submitted proposals that were reviewed.

Accordingly, for the foregoing reasons and on this preliminary record, Patent Owner’s evidence of skepticism of others is weak.

(vi) Copying

“Copying may indeed be another form of flattering praise for inventive features.” *Crocs, Inc. v. ITC*, 598 F.3d 1294, 1311 (Fed. Cir. 2010). Copying “requires evidence of efforts to replicate a specific product.” *Wyers v. Master Lock Co.*, 616 F.3d 1231, 1246 (Fed. Cir. 2010). “This may be demonstrated either through internal documents; direct evidence such as disassembling a patented prototype, photographing its features, and using the photograph as a blueprint to build a virtually identical replica; or access to, and substantial similarity to, the patented product (as opposed to the patent).” *Iron Grip Barbell Co. v. USA Sports, Inc.*, 392 F.3d 1317, 1325 (Fed. Cir. 2004) (internal citations omitted). “We note, however, that a showing of copying is only equivocal evidence of nonobviousness in the absence of more compelling objective indicia of other secondary considerations.” *Ecolochem, Inc. v. S. Cal. Edison Co.*, 227 F.3d 1361, 1380 (Fed. Cir. 2000); *see also In re GPAC*, 57 F.3d 1573, 1580 (Fed. Cir. 1995) (“[M]ore than the mere fact of copying by an accused infringer is needed to make that action significant to a determination of the obviousness issue.” (quoting *Cable Elec. Prods. v. Genmark, Inc.*, 770 F.2d 1015, 1028 (Fed. Cir. 1985))); *Institut Pasteur & Université Pierre et Marie Curie v. Focarino*, 738 F.3d 1337, 1347–48 (Fed. Cir. 2013) (“Copying requires duplication of features of the patentee’s work based on access to that work, lest all infringement be mistakenly treated as copying. . . . But the Board did

IPR2021-00108
Patent 8,676,877 B2

not analyze whether Pasteur’s showing of the similarities of its method to the content of the cited publications, *e.g.*, their use of the same specific GIIIE endonuclease, indicated that the publications’ authors had access to, and borrowed from, the Pasteur sources.”); *Liqwd, Inc. v. L’Oreal USA, Inc.*, 941 F.3d 1133, 1136–39 (Fed. Cir. 2019) (mere access to information coupled with allegations of infringement are not sufficient evidence of copying, absent evidence of other circumstances, such as copying specifics of a disclosed or actual product, or altering a design after obtaining access to the information). “Of course, the proponent of objective evidence offered to show nonobviousness, such as copying, must show that a nexus exists between the evidence *and the claimed features of the invention.*” *Liqwd*, 941 F.3d at 1138 (emphasis added).

Patent Owner asserts that it provided information to Petitioner regarding its SET product during a November 2012 meeting and in a subsequent “email attaching a specially tailored data sheet.” Prelim. Resp. 45–46 (citing Ex. 2036 ¶ 56). Patent Owner argues that “Petitioner then copied the claimed invention by incorporating the claimed architecture into Petitioner’s GPGPUs in the manner described by the datasheet” and “named its GPU interconnect architecture, which uses the claimed structure, NVLink™.”¹³ *Id.* at 46 (citing Ex. 2036 ¶ 59); *see also* PO Sur-reply 9–10.

Petitioner traverses Patent Owner’s assertions of copying, calling it “a gross misrepresentation to the PTAB.” Pet. Reply 8–9. Petitioner asserts

¹³ Dr. Dauger refers to NVIDIA’s “general-purpose GPU (‘GPGPU’) supercomputing” and describes “NVIDIA’S Tesla GPGPUs as hardware black boxes on which the back end executes.” Ex. 2036 ¶¶ 56–57 (citing Ex. 2020, 4, Fig. 3).

IPR2021-00108
 Patent 8,676,877 B2

that the inventors of the '877 patent sent an unsolicited email to one of its employees attaching a public SET datasheet that “did not have any suggestion of ‘provid[ing] a communications infrastructure for direct all-to-all communications between each GPU.’” *Id.* at 8 (alteration by Petitioner) (citing Prelim. Resp. 56).

Patent Owner provides no evidence to support its assertion. Patent Owner does not provide any description of the NVLink product or compare this product to the claims of the '877 patent. On this record, Patent Owner's conclusory assertions are inadequate to establish copying of the claimed invention by Petitioner.

Accordingly, for the foregoing reasons and on this preliminary record, Patent Owner's evidence of copying is weak.

d. Summary

Based on the foregoing discussion, Petitioner sufficiently establishes for purposes of institution that the combination of Schreiner1, Schreiner2, Schreiner3, the Maple Guide, PC Magazine1, and PC Magazine 2, with or without the Distributed Maple Code, renders claim 1 and 8–11 obvious.

Accordingly, Petitioner establishes a reasonable likelihood of prevailing with respect to claims 1 and 8–11.

E. Alleged Obviousness of Claims 1, 10, and 11 over Schreiner1, Schreiner2, Schreiner3, Maple Guide, Distributed Maple Code, PC Magazine, PC Magazine 2, and Nayak

1. Nayak (Ex. 1031)

Nayak teaches the use of digital signal processors (DSP) for executing MATLAB based software. Ex. 1031, 1 (Abstract), Fig. 4.

IPR2021-00108
 Patent 8,676,877 B2

2. *Alleged Obviousness for Claims 1, 10, and 11*

Petitioner contends claims 1, 10, and 11 would have been obvious over the combination of Schreiner1, Schreiner2, Schreiner3, Maple Guide, Distributed Maple Code, PC Magazine, PC Magazine 2, and Nayak. *See* Pet. 71–76.

Petitioner relies on Nayak further to suggest using a special purpose microprocessor in Schreiner1’s system. *See* Pet. 72–77. Petitioner contends that Nayak teaches “an MPI cluster-based approach for executing MATLAB mathematics software that is similar to Maple and Mathematica.” *Id.* at 71. (citing Ex. 1031, 1 (Abstract)). Petitioner contends that “Nayak takes account of the fact that MATLAB, like Maple and Mathematica, is an interpreter, and teaches an approach that a POSITA would find obvious to apply to the Distributed Maple Publications in order to deploy Distributed Maple interpreter kernels on DSP cluster nodes.” *Id.* at 75 (citing Ex. 1005 ¶ 163). Petitioner also contends that Nayak teaches advantages of implementing matrix operations on DSPs, and it would have been obvious to implement Schreiner1’s ssiPlot matrix operation functions in the manner taught by Nayak.” *Id.* at 74 (citing Ex. 1031, section 5; Ex. 1008, 32; Ex. 1005 ¶ 161). Petitioner provides record citations to evidence, including the testimony of Dr. Tufo, to support its showing. *See id.* at 72–77.

Based on the foregoing discussion and a review of the record, for purposes of institution, Petitioner sufficiently shows that the combination of Schreiner1, Schreiner2, Schreiner3, Maple Guide, Distributed Maple Code, PC Magazine, PC Magazine 2, and Nayak renders obvious the subject matter of claims 1, 10, and 11. Other than presenting arguments as addressed

IPR2021-00108
Patent 8,676,877 B2

above with respect to claims 1 and 10, Patent Owner does not address this alternative showing by Petitioner of claims 1, 10, and 11 based on Nayak.

Nonetheless, the burden remains on Petitioner to demonstrate unpatentability. *See Dynamic Drinkware, LLC v. Nat'l Graphics, Inc.*, 800 F.3d 1375, 1378 (Fed. Cir. 2015)

Accordingly, Petitioner establishes a reasonable likelihood of prevailing with respect to claims 1, 10, and 11.

F. Alleged Obviousness of Claim 5 over Schreiner1, Schreiner2, Schreiner3, Maple Guide, Distributed Maple Code, PC Magazine, PC Magazine 2, Nayak, and Kelly

Petitioner adds Kelly to the references addressed in the previous section to address claim 5. *See* Pet. 76–77. Claim 5 depends from claim 1 and recites “wherein the special purpose microprocessor comprises multiple processor cores.” Petitioner provides record citations to evidence, including the testimony of Dr. Tufo, to support its showing. *See id.*

Based on a review of the record, for purposes of institution, Petitioner sufficiently shows that the combination of Schreiner1, Schreiner2, Schreiner3, Maple Guide, Distributed Maple Code, PC Magazine, PC Magazine 2, Nayak, and Kelly renders obvious the subject matter of dependent claim 5. Patent Owner does not address claim 5 separately. Nonetheless, the burden remains on Petitioner to demonstrate unpatentability. *See Dynamic Drinkware*, 800 F.3d at 1378.

Accordingly, Petitioner establishes a reasonable likelihood of prevailing with respect to claim 5.

IPR2021-00108
Patent 8,676,877 B2

G. *Alleged Obviousness of Claim 5 over Schreiner1, Schreiner2, Schreiner3, Maple Guide, Distributed Maple Code, PC Magazine, PC Magazine 2, the SPARC IV Article, and the AMD Article*

Petitioner alternatively employs the SPARC IV Article and the AMD Article to the references employed against independent claim 1 to support its showing of alleged obviousness of dependent claim 5. *See* Pet. 77–80.

Petitioner provides record citations to evidence, including the testimony of Dr. Tufo, to support its showing. *See id.*

Based on a review of the record, for purposes of institution, Petitioner sufficiently shows that the combination of Schreiner1, Schreiner2, Schreiner3, Maple Guide, Distributed Maple Code, PC Magazine, PC Magazine 2, the SPARC IV Article, and the AMD Article, renders obvious the subject matter of dependent claim 5. Patent Owner does not address claim 5 separately. Nonetheless, the burden remains on Petitioner to demonstrate unpatentability. *See Dynamic Drinkware*, 800 F.3d at 1378.

Accordingly, Petitioner establishes a reasonable likelihood of prevailing with respect to claim 5.

H. *The Petition's Word Limit*

Patent Owner alleges that the Petition “undercount[s] the true word count” by “about 900 excess words” because it uses “non-standard formats such as ‘EX-1001,’ ‘Schreiner1,’ and ‘¶216’ to make two words count as one and cop[ies] text from the alleged prior art as images on pages 22, 24, 52, and 55–58 of the Petition to avoid quoting text and counting the words.” Prelim. Resp. 54 (citing *Starbucks Corp. v. Ameranth, Inc.*, CBM2015-00091, Paper 16 at 2–3 (PTAB Jan. 29, 2016)). It is not clear if by “900 excess words,” Patent Owner implies that the total word count would be 14,900, or if it would be 900 words in excess of the certified word count of

IPR2021-00108
 Patent 8,676,877 B2

13,882. *See id.* (noting “37 C.F.R. § 42.24(a)(i) limits the Petition to 14,000 words” and “[t]he Petition certifies its word count is 13,882”).

In any case, the allegedly non-standard formats and copied text from images do not result in an excessive word count or violate the spirit of the word count underlying the rule. For example, the Petition reproduces a “screenshot of the Maple text interface” on a user interface in Schreiner1, but Petitioner simply could have cited the pages to make the same point. *See* Pet. 21 (reproducing Ex. 1008, 7–8). Petitioner again reproduces most of the same figure on page 24, but only to illustrate that the screenshot represents a portion of a user interface in Schreiner. The information carried in the text is not reproduced for what the text actually represents, other than to show it represents code “entered by the user after each ‘>’ prompt.” *See id.* at 24. Similar remarks apply to the screenshot on page 52 of the Petition. *Id.* at 52 (“Schreiner 1 shows a screenshot of a session in which the user inputs several instructions . . .”). The reproduced snapshots on pages 55–57 simply show computer code and not much of it (i.e., about 9 or 10 “words” of code on each page except page 57, which shows about 20 “words” of code). Counting computer code as words that violate the Petition’s word limits in each image of code, where Petitioner repeats the same images of code, is not reasonable in the context of Petitioner’s showing here, because each “word” of code primarily carries meaning by association with the code words that surrounds it, much like an image.

Patent Owner does not cite a case in which the Board granted relief based on the use of such alleged non-standard formats for citations. Patent Owner also does not specify any form of relief that it seeks. *See id.*

IPR2021-00108
Patent 8,676,877 B2

Nevertheless, the Board hereby authorizes both parties to use the allegedly non-standard citation formats in future papers in this trial.

III. CONCLUSION

After considering the evidence and arguments presented in the Petition and the Preliminary Response and additional briefing, we determine Petitioner has demonstrated a reasonable likelihood that it would prevail with respect to all of its unpatentability challenges and the challenged claims. We institute an *inter partes* review on the challenged claims and all of the grounds presented in the Petition. At this stage of the proceeding, we have not made a final determination as to the patentability of these challenged claims.

IV. ORDER

Accordingly, it is

ORDERED that pursuant to 35 U.S.C. § 314, *inter partes* review is instituted as to the challenged claims of the '877 patent with respect to all grounds of unpatentability presented in the Petition; and

FURTHER ORDERED that *inter partes* review is commenced on the entry date of this Order, and pursuant to 35 U.S.C. § 314(c) and 37 C.F.R. § 42.4, notice is hereby given of the institution of a trial.

IPR2021-00108
Patent 8,676,877 B2

For PETITIONER:

Brent Yamashita
Jonathan Hicks
DLA PIPER LLP
brent.yamashita@dlapiper.com
jonathan.hicks@dlapiper.com

For PATENT OWNER:

Jon W. Gurka
Ted M. Cannon
Cheryl T. Burgess
KNOBBE MARTENS OLSON AND BEAR LLP
2jwg@knobbe.com
2tmc@knobbe.com
2ctb@knobbe.com

Exhibit 5

Supercomputing Engine for *Mathematica*

Supercomputing 2008
Austin, Texas



Dauger Research



Advanced Cluster Systems

Supercomputing Engine for Mathematica

Dean E. Dager, Ph. D.
President, Dager Research, Inc.
d@dagerresearch.com

The First Mac Cluster

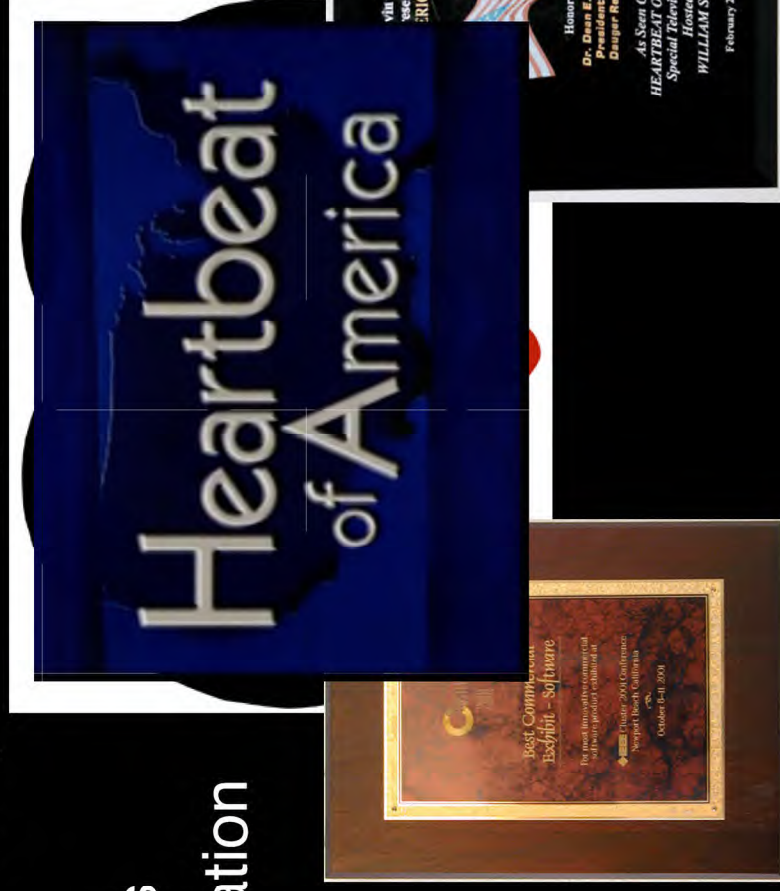
- established 1998



The Dawson Cluster

High-Performance, Scientific, and Cluster Computing

- Software
 - “Plug-and-Play” Supercomputer-Compatible Clusters
 - Pooch Application
 - Source-Code Tutorials
 - Visualization & Simulation
- Consulting Services
 - Optimization
 - Parallelization
 - Vectorization



Why Parallel Computing?

Problems too large to solve on one computer

- Takes too much time
- Requires more memory
 - can outgrow RAM capacity

Programming API standardized

- Message-Passing Interface (MPI)
 - specification established in 1994
 - dominant software interface at supercomputing centers
 - portable MPI code in Fortran and C



Why Parallel Computing?

Problems too large to solve on one ~~computer~~ “core”

- Computational power *per core* no longer doubles
- Doubling cores is how Moore’s Law technically holds
- Multicore utilization is not automatic
- Software must make up for where hardware leaves off
 - Choose your parallel programming paradigm wisely

Multicore Eroding Moore’s Law

- http://macresearch.org/multicore_eroding_moorees_law



MPI in *Mathematica*



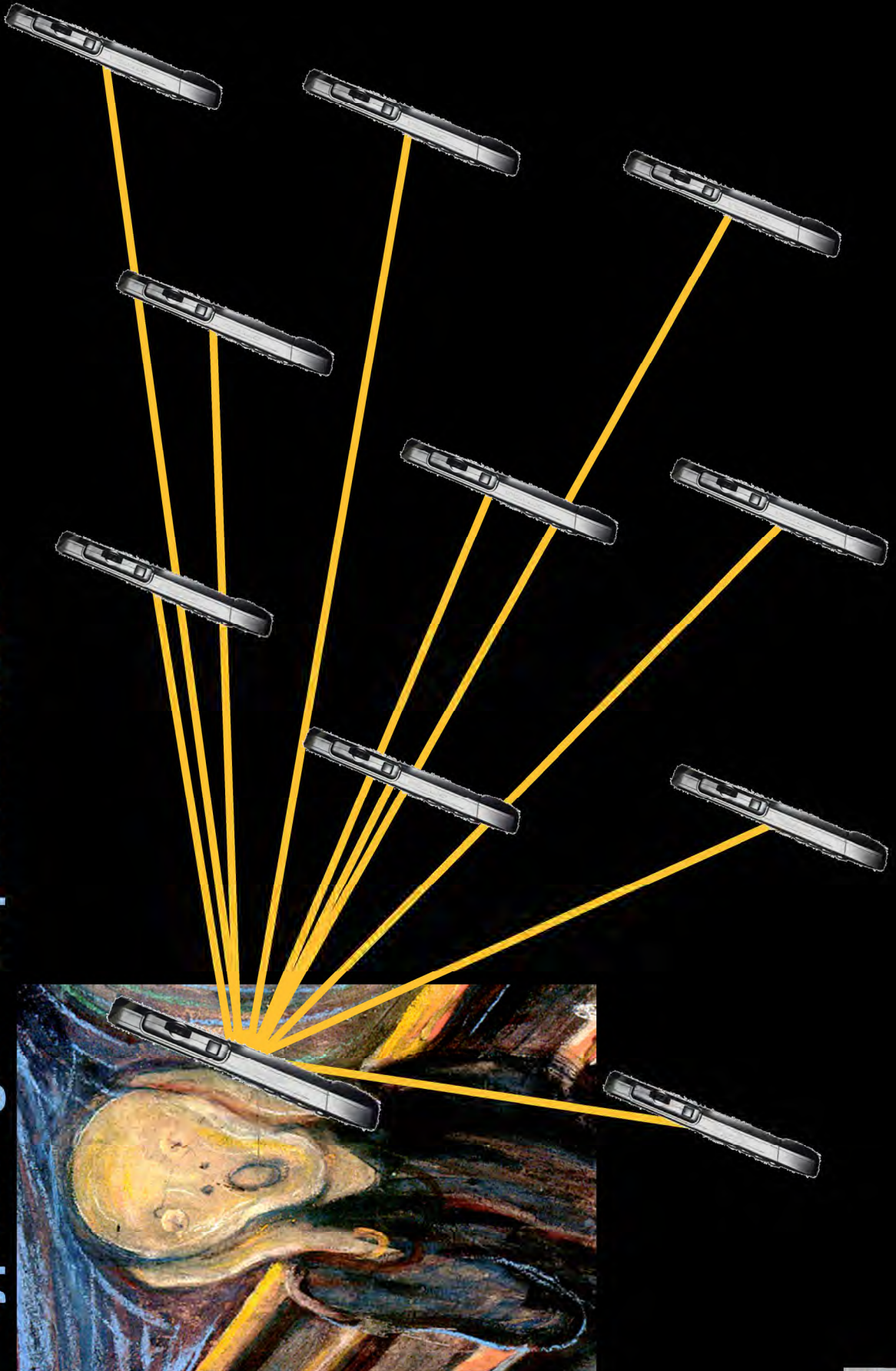
Supercomputing Engine for *Mathematica*

- Closely follows MPI standard in *Mathematica* environment
 - Basic MPI calls (mpiSend, mpiRecv)
 - Asynchronous MPI calls (mpiSend, mpiRecv, mpiTest)
 - Collective MPI calls (mpiBcast, mpiGather, mpiAlltoall)
- High-level parallel calls for common tasks
 - ParallelTable, EdgeCell, ParallelFourier, ElementManage
- Basic Parallel I/O
- Automatically locates, launches, configures, and coordinates *Mathematica* kernels via Pooch
 - from command line or *Mathematica*'s Front End
- Builds on any licensed grid*Mathematica*



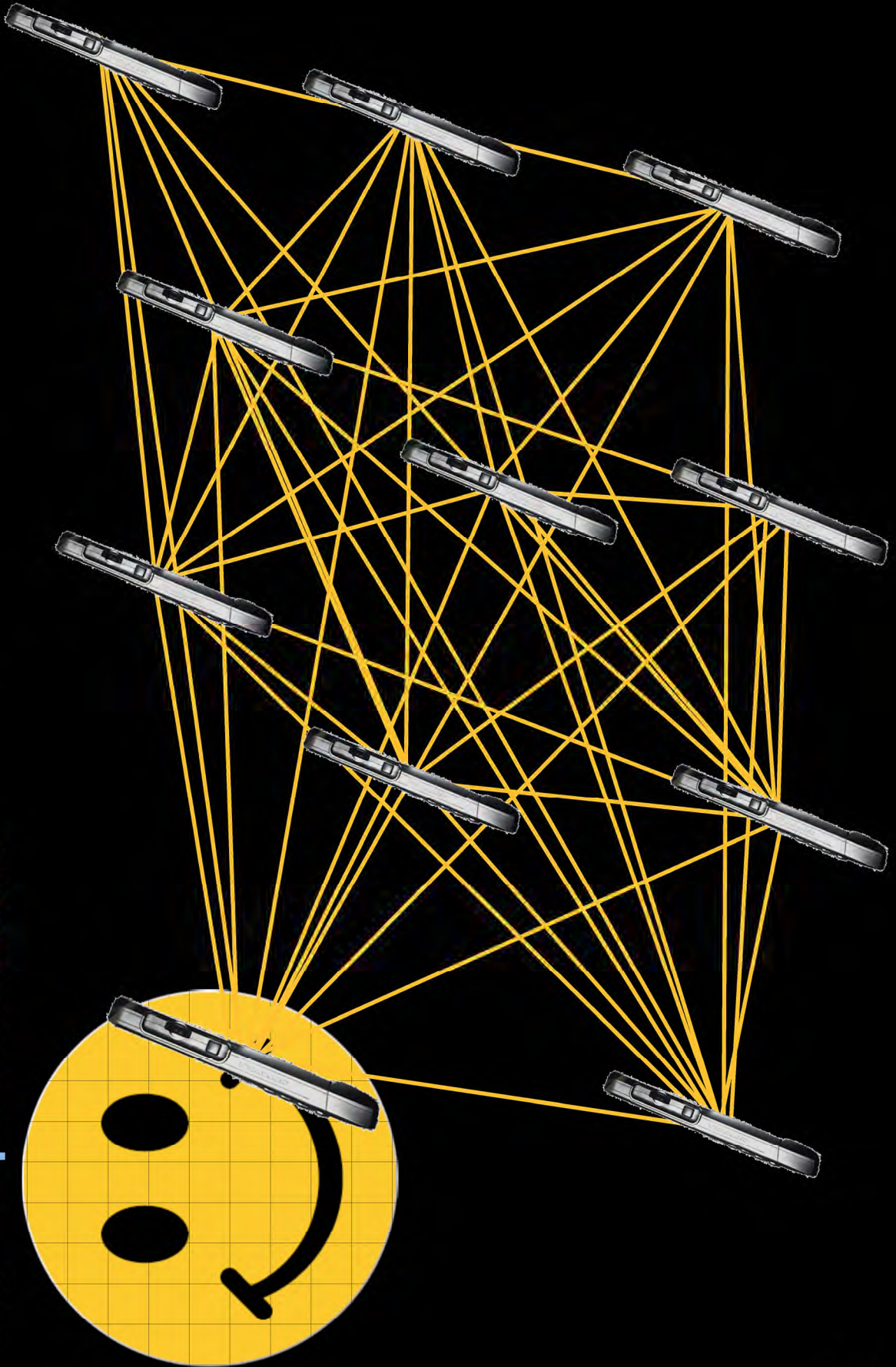
Why MPI instead of “grid”?

Typical “grid” implementation



Why MPI instead of “grid”?

MPI implementation



Parallel Code using MPI

Code coordinating parallel work using messages

- N tasks or “virtual processors” running simultaneously, labeled 0 through N-1
- executables often use identification data to determine algorithmically what part of the problem on which to work
- tasks pass messages amongst themselves to organize data and coordinate work
- any group of tasks can communicate ($\Rightarrow O(N^2)$ connections)
 - simple sends and receives
 - collective calls - broadcast, gather, reduce, transpose, etc.
- synchronization not required, but often implied by messages

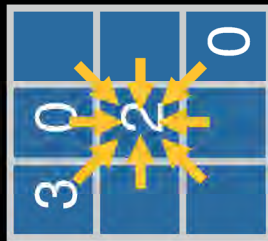


Parallel “Life”

Message-Passing for Cellular Automata

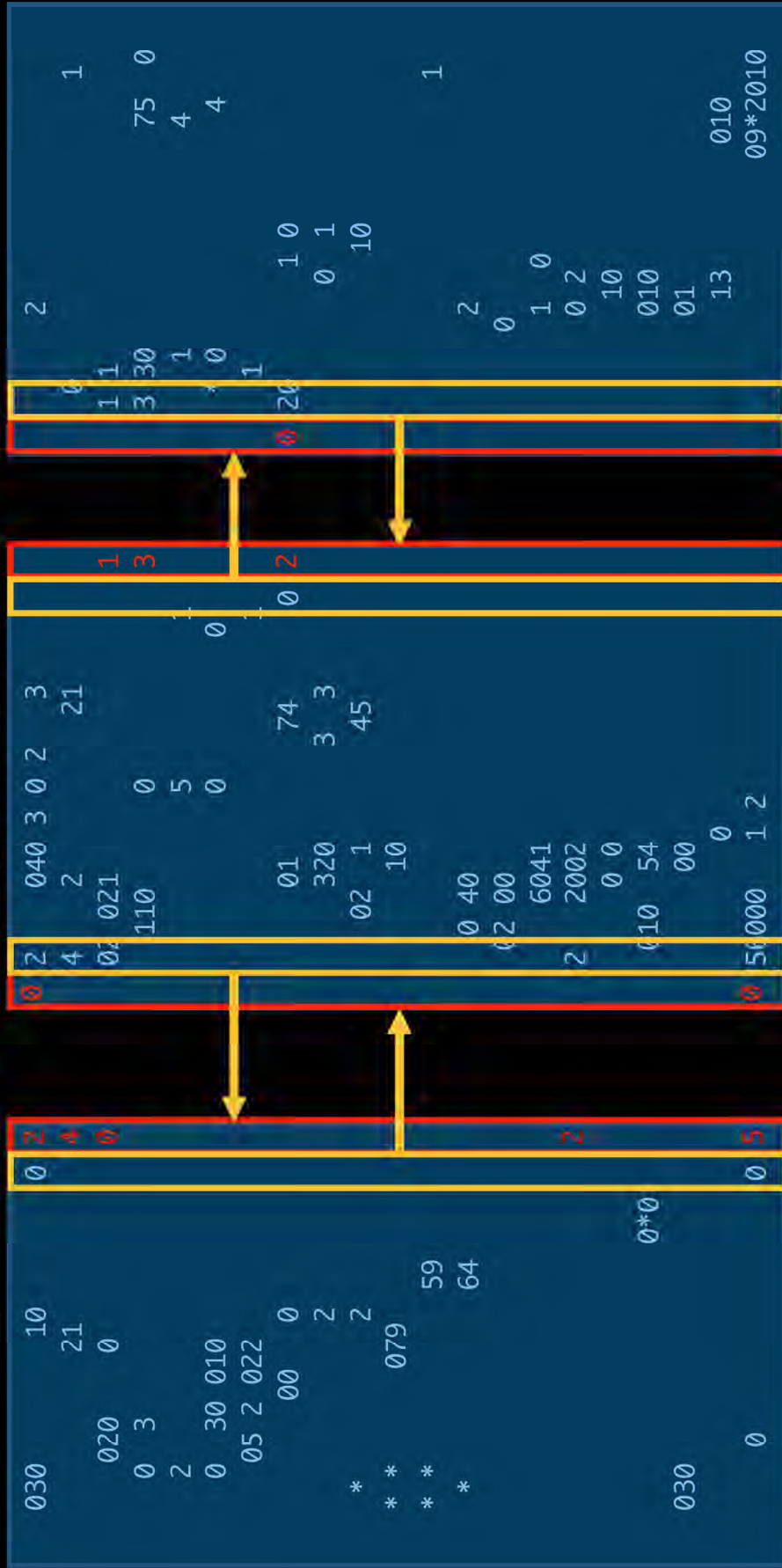


Data to propagate between partitions



Parallel “Life”

Message-Passing for Cellular Automata

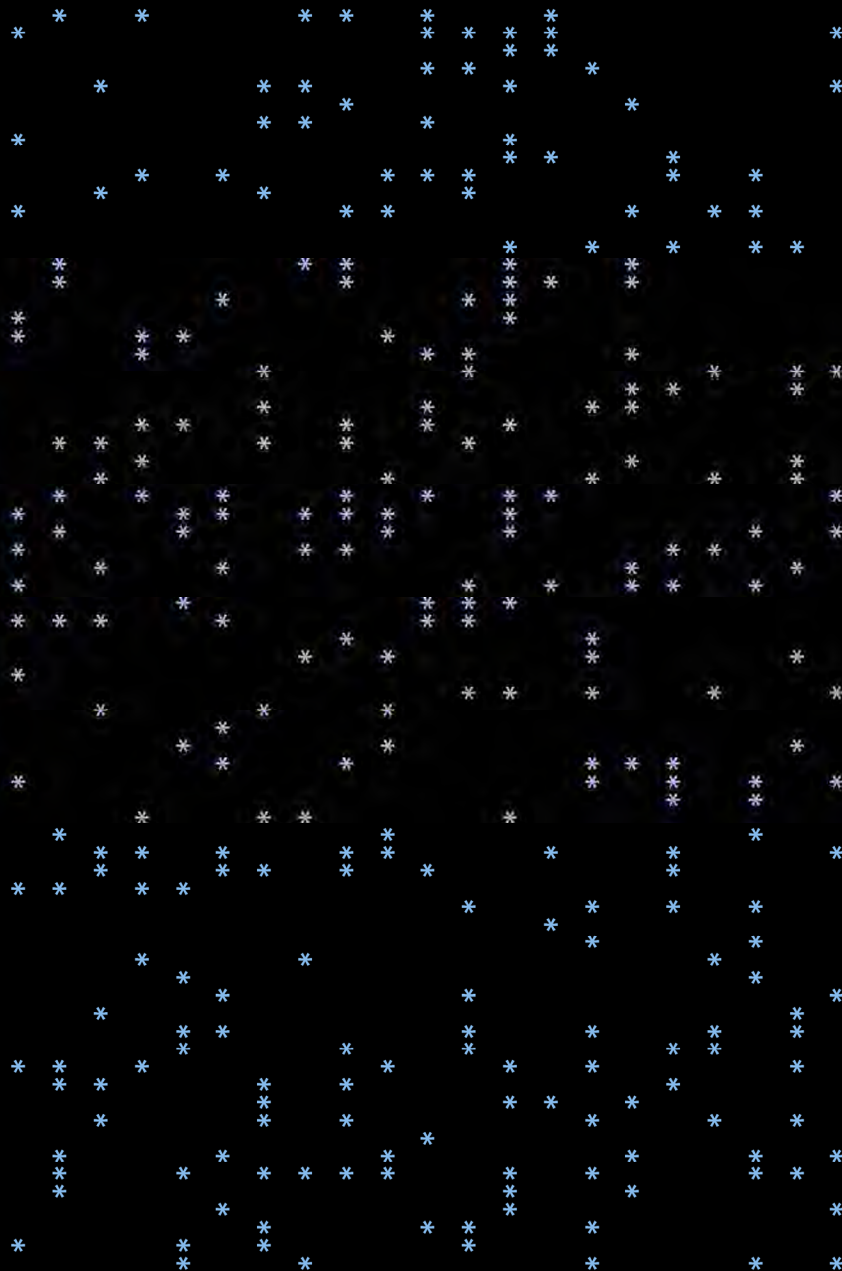


Message exchange maintains “guard cells”



Plasma Simulation

Plasma Dynamics

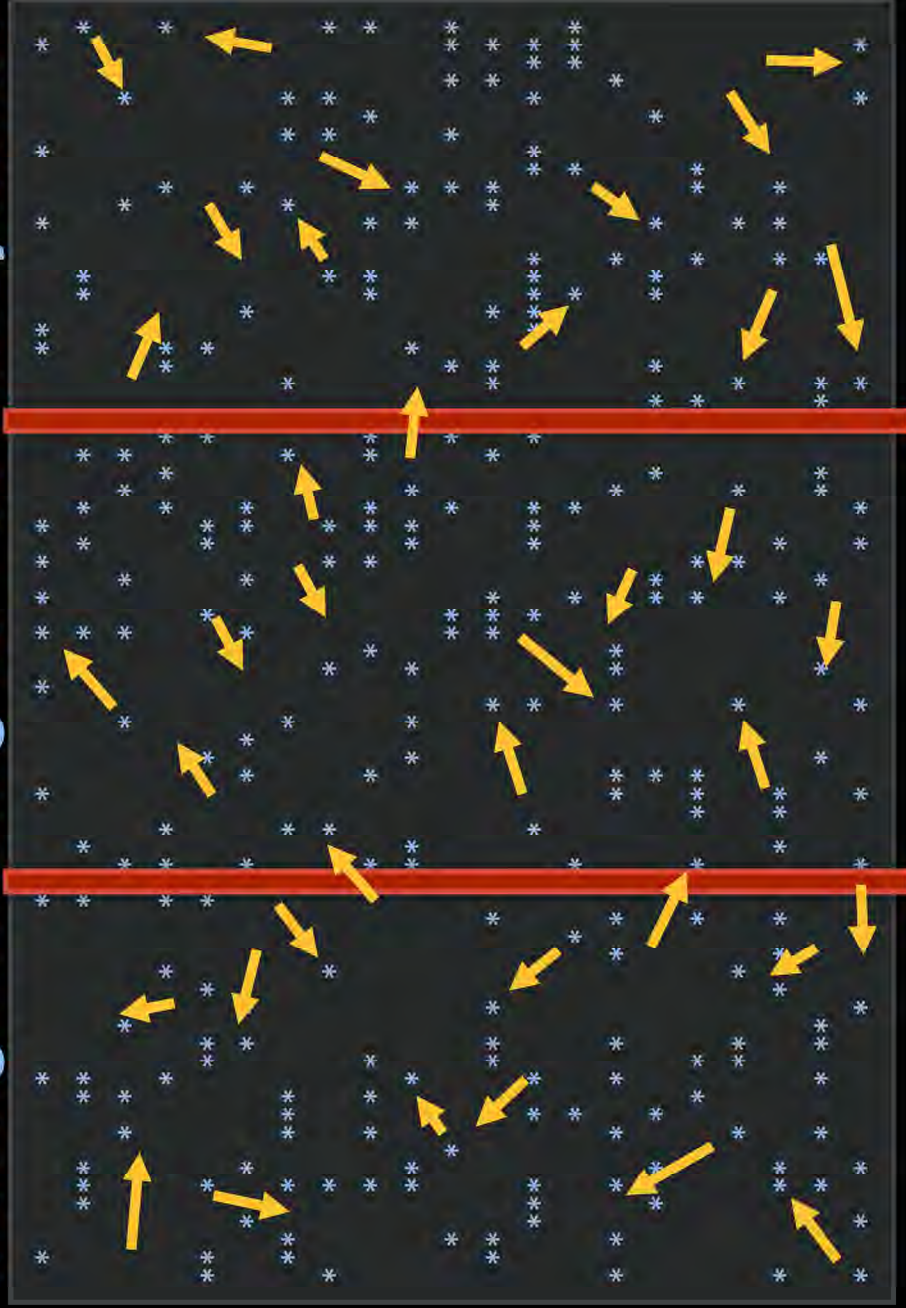


V. K. Decyk, C. D. Norton, *Comp. Phys. Communications* **164** (2004) 80-85



Parallel Plasma Simulation

Message-Passing for Plasma Dynamics

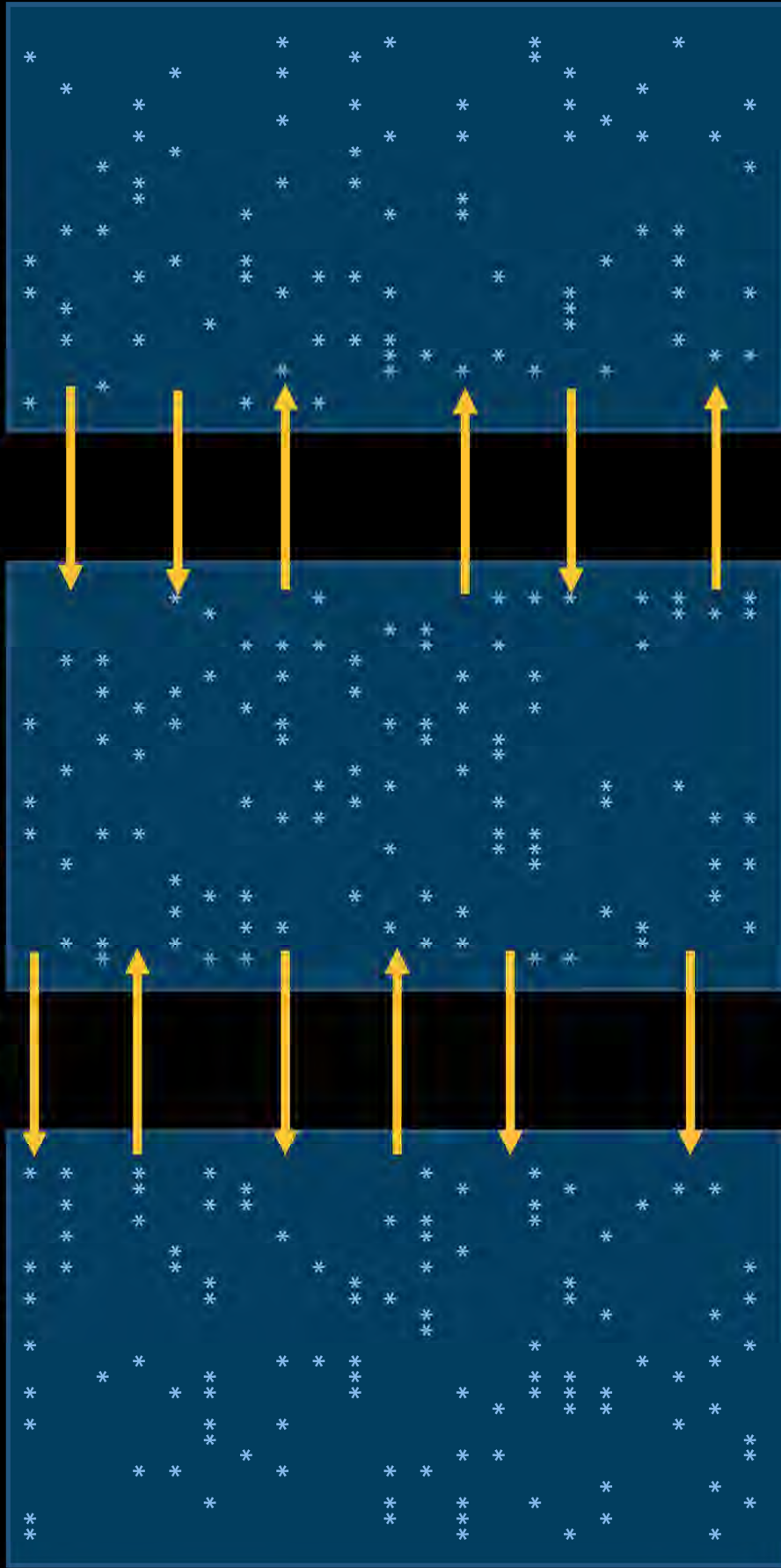


V. K. Decyk, C. D. Norton, *Comp. Phys. Communications* **164** (2004) 80-85



Parallel Plasma Simulation

Message-Passing for Plasma Dynamics

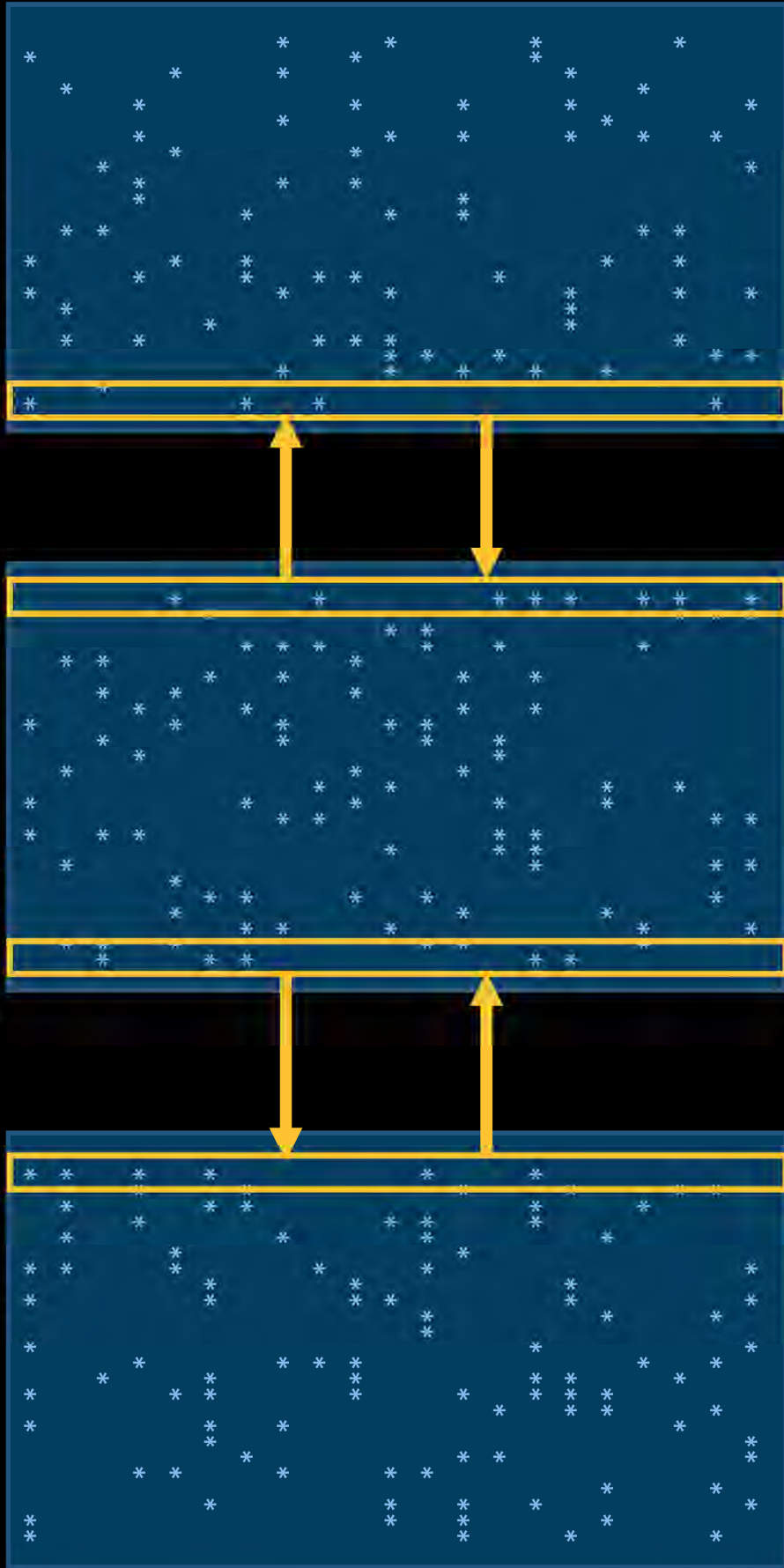


Particles propagate from one partition to its neighbor



Parallel Plasma Simulation

Message-Passing for Plasma Dynamics

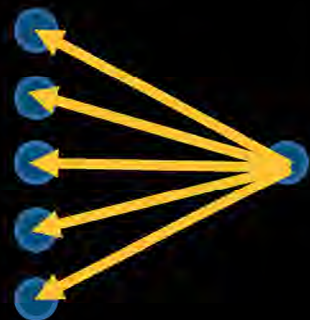


Particles propagate from one partition to its neighbor



Message-Passing Patterns

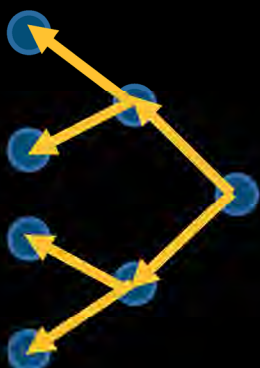
Supported via MPI



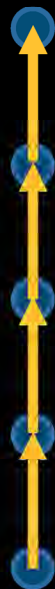
“Master-Slave”



All to All



Tree



Nearest Neighbor



Irregular

Or Combinations



A Demonstration

Dean E. Dager, Ph. D.
d@daugerresearch.com



Advanced Cluster Systems

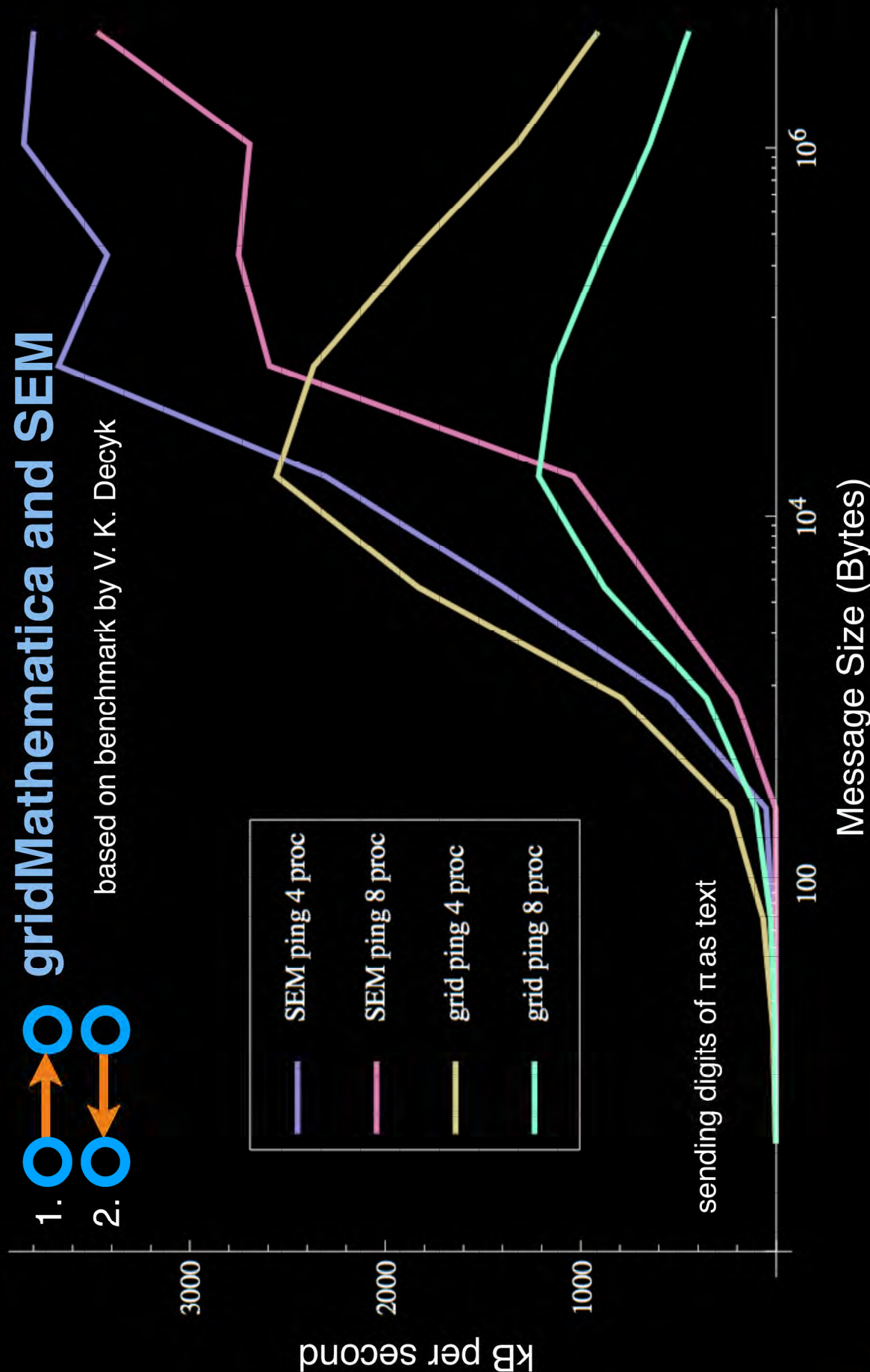
<http://daugerresearch.com>

Ping-pong Benchmark

1. 
2. 

gridMathematica and SEM

based on benchmark by V. K. Decyk

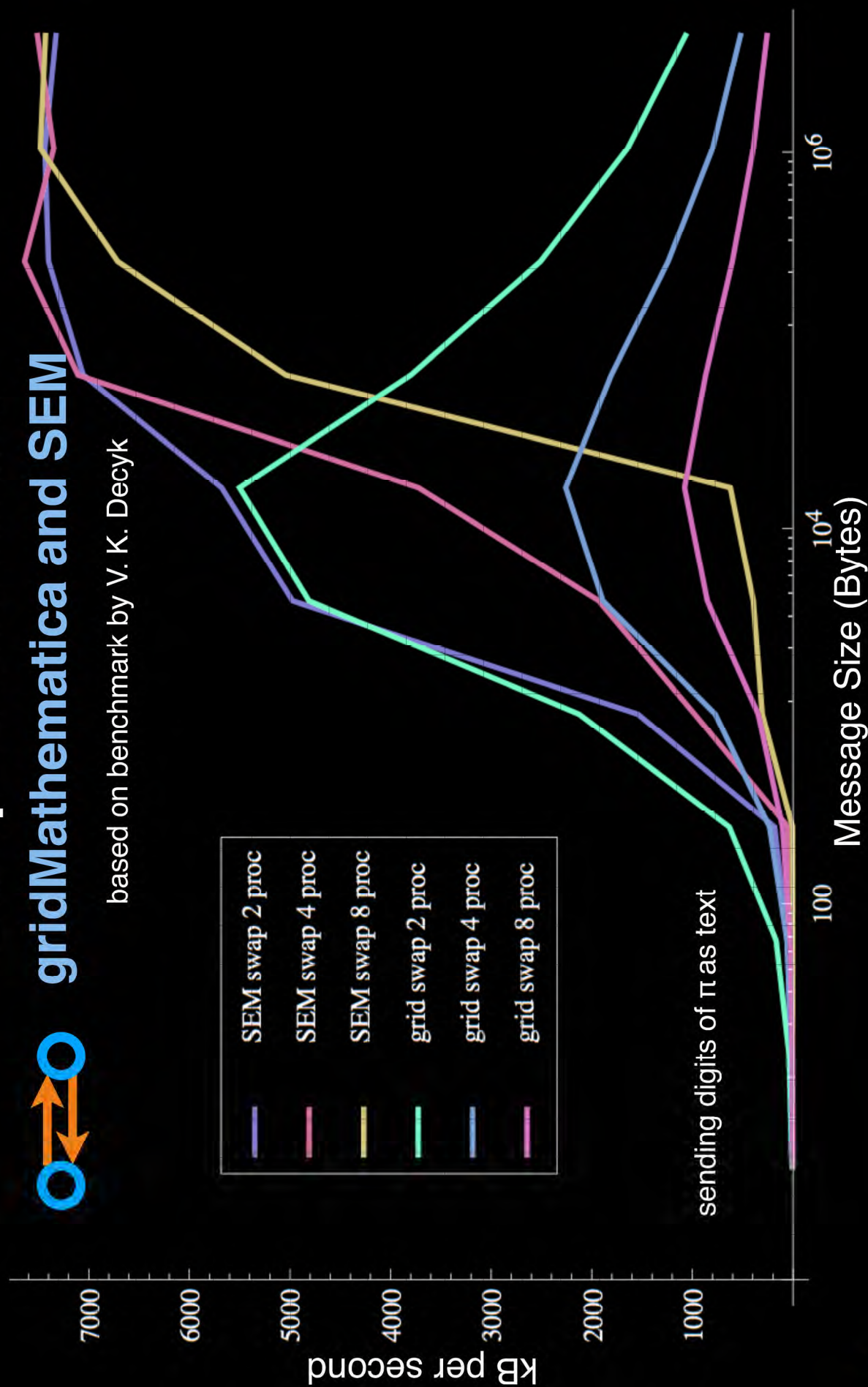


Swap Benchmark



gridMathematica and SEM

based on benchmark by V. K. Decyk



“We found SEM to be very efficient in terms of stability and use for financial engineers who do not have the time to optimize load balancing issues but want to focus on modeling.”

- Vineer Bhansali, Managing Director,
PIMCO



Dauger Research



Advanced Cluster Systems

"I CAN endorse SEM with pride. I would emphasize the flexibility and scalability of SEM. And SEM can make writing *Mathematica* programs even more flexible than before."

- *Yuko Matsuda, Professor,
Tokyo Institute of Technology*



Dauger Research



Advanced Cluster Systems

Who to Contact

Dean E. Dager, Ph. D.
Dager Research, Inc.
d@dangerresearch.com

Zvi Tannenbaum
Advanced Cluster Systems, LLC
zvi@advclustersys.com



<http://dangerresearch.com> & <http://advclustersys.com>

Roadmap

Wolfram Research Booth

**Booth #163
Wed 3:00 pm**

Exhibitor Forum Session

**Room 19AB
Thurs 10:30 am**

Advanced Cluster Systems Booth

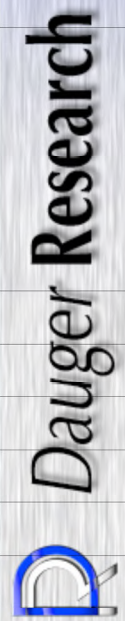
**Booth #162
Mon-Thurs**

Machine Evaluation Workshop 19

**Daresbury, UK
2 Dec 2008**



For More Information



Reference Library

Documentation

Supercomputing Engine for *Mathematica* Site

<http://daugerresearch.com/pooch/mathematica/>

Advanced Cluster Systems Site

<http://advclustersys.com/>

Tutorials on Writing Parallel Code

<http://daugerresearch.com/vault/tutorials/>

Mac Clustering on National Television

<http://daugerresearch.com/awards/KeepingAmericaStrong/>

Related Publications

<http://daugerresearch.com/vault/publication/>

Multicore Eroding Moore's Law

http://macresearch.org/multicore_eroding_moorees_law



Q&A

Dean E. Dager, Ph. D.
President, Dager Research, Inc.
d@dagerresearch.com



Dager Research



Advanced Cluster Systems

<http://dagerresearch.com> & <http://advclustersys.com/>

EXHIBIT 6

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

NVIDIA CORPORATION,
Petitioners,

v.

ADVANCED CLUSTER SYSTEMS, INC.
Patent Owner

Case No. IPR2021-00020
U.S. Patent 10,333,768

DECLARATION OF JASWINDER PAL SINGH, Ph.D.

NVIDIA Corp. v. Advanced Cluster Systems IPR2021-00020 Advanced Cluster Systems Ex. 2001

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

1. I, Jaswinder Pal Singh, Ph.D., am making this declaration at the request of Patent Owner Advanced Cluster Systems, Inc. (“ACS”) in the matter of the *Inter Partes* Review No. IPR2021-00020 of U.S. Patent No. 10,333,768 (“the ’768 patent”). I understand that this declaration is being submitted in the IPR as Exhibit 2001.

2. I am being compensated for my work in this matter at my standard hourly rate for consulting services. My compensation in no way depends on the outcome of this proceeding.

3. In conducting the analysis and forming the opinions set forth in this Declaration, I considered, in addition to my knowledge and expertise in the relevant field, at least the following written materials:

- ’768 patent (Ex. 1101);
- File history of the ’768 patent (Ex. 1102);
- NVIDIA’s Petition (Paper 1);
- Declaration of Dr. Henry Tufo (Ex. 1105);
- Declaration of Dr. Wolfgang Schreiner (Ex. 1106);
- Schreiner 1 (Ex. 1108);
- Schreiner 2 (Ex. 1109);
- Schreiner 3 (Ex. 1110);
- Maple Guide (Ex. 1111); and

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

- Any other document referenced herein.

PROFESSIONAL BACKGROUND

4. I am currently a Professor of Computer Science at Princeton University, a position I have held for more than 15 years. Before becoming a Full Professor, I was an Assistant, and then Associate, Professor at Princeton.

5. I received a Bachelor's Degree in Electrical Engineering and Computer Science, *summa cum laude*, from Princeton in 1987. I received an M.S. Degree in Electrical Engineering in 1989 and a Ph.D. Degree in Electrical Engineering in 1993, both from Stanford University. For my Ph.D. thesis, I researched the boundary of applications and parallel computer systems, including programming models, and the interactions between software and hardware. I understand that my current curriculum vitae (CV) is being submitted in this proceeding as Exhibit 2002.

6. I joined the faculty at Princeton in 1995, where I served as an Assistant Professor in the Computer Science Department. My initial research area included parallel computer systems and applications with a focus on (i) scalable multiprocessing in both the shared address space and message passing paradigms, including studying the programming and performance tradeoffs between the two, (ii) understanding how shared address space abstractions might be supported using commodity parts, and how applications might be restructured for such systems, and

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

(iii) the use of high-performance computing in biology, medicine, cosmology, and other areas.

7. From 1999 to 2010 I was the director of Princeton's "Program in Integrative Computer and Application Sciences" (PICASso), a multi-department, university-wide interdisciplinary program focused on scalable parallel and distributed computing at the boundary of computer science and a broad range of application sciences. Directing that program required me to develop and foster new interdisciplinary courses, new practice-oriented workshops in scalable computing based on needs gathered from multiple departments, new interdisciplinary seminar series, and programs in effective written and oral scientific communication. I oversaw the creation of courses for an interdisciplinary computational science curriculum, taught by faculty in different departments, in addition to parallel and distributed computing courses. From 2003 to the present, I have been on the Executive Committee at Princeton's Institute for Computational Science and Engineering (PICSSiE). That permanent institute came out of the PICASso program that I led, and the Institute has taken over PICASso's activities as well, including a permanent Graduate Certificate Program in Computational Science and Engineering at Princeton that I initially developed.

8. In addition to my concurrent work as a professor, beginning in 1999, I was a member of the Technical Advisory Board at NOAA Geophysical Fluid

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

Dynamics Laboratory. In that role I aided in the shift from vector to parallel computing, and in the procurement of large parallel computing and analysis infrastructures.

9. At Princeton, I have taught many courses in areas relevant to the technology disclosed in the '768 patent. These include undergraduate and graduate courses in parallel computing, parallel architecture and programming, scalable infrastructure and services, and in analysis of data. I taught several new interdisciplinary courses I created for the PICASSo program regarding parallel and distributed computing. I also created and taught courses in Marketplace Design and at the boundary of technology, business, marketplaces, and economics, including one which has been dubbed the "CTO Course."

10. In the course of my work at Princeton, I have supervised more than 15 students engaged in thesis research projects in support of Master of Science and Doctor of Philosophy degrees. Those projects have included topics such as new programming models for scalable parallel computing, peer-to-peer publish-subscribe systems, and a comparison of programming models for high-performance parallel computing.

11. I am the author of a leading textbook on parallel computing, "Parallel Computer Architecture: A Hardware-Software Approach," by David E. Culler,

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

Jaswinder Pal Singh with Anoop Gupta, Morgan Kaufmann Publishers, 1999. I have also published more than 100 refereed conference and journal publications.

12. I am a member of the Association for Computing Machinery and the Institute of Electrical and Electronics Engineers.

13. I have been recognized for my research and other professional endeavors. I was honored with a Presidential Early Career Award for Scientists and Engineers (PECASE), 1997, awarded to twenty young scientists and engineers in the United States selected from all areas of science/engineering by the National Science Foundation, and the Sloan Research Fellowship, 1998, awarded to ten Computer Scientists nationwide by the Alfred Sloan Foundation. Also, my 2005 publication for the International Distributed Event-based Systems (DEBS) was in 2019 voted as the most influential paper of DEBS 2005.

14. My professional activities further include serving on corporate boards of directors, including the companies 8x8, Inc. and Hiro, PBC (formerly Blockstack, PBC).

15. I have accepted invitations to evaluate research and education programs and to evaluate proposals, including for the US Government (NOAA's procurement programs and the National Science Foundation) and the Swedish Government (Evaluating the ARTES/PAMP national program for scientific research and graduate education in high-performance computing, as well as other programs for the Swedish

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

Research Council). I have also served as an external evaluator for faculty hiring at Uppsala University, Sweden and the University of Copenhagen, Denmark. I have delivered many invited presentations and lectures at a variety of international and domestic venues with details provided in my curriculum vitae.

16. I have served on the program committees and as program chair and track chair of many ACM, IEEE and other conferences in scalable computing, including the International Symposium on Computer Architecture, Supercomputing/SC, SIGMETRICS Conf. on Measurement and Modeling of Computer Systems, Principles and Practice of Parallel Programming, Symposium on Parallel Algorithms and Architectures, International Conference on Supercomputing, International Parallel Processing Symposium, International Conference on Parallel Processing etc., as well as several ACM and IEEE workshops on scalable computing, data analysis and mining, architecture, and languages and compilers.

17. In my education, research, and consulting work, I have used and designed many systems for parallel and clustered computation.

18. I am a named inventor of six issued patents, including U.S. Patent Nos. 7,080,073; 6,915,294; 7,415,469; 7,103,838; 10,796,276; and 10,796,277.

19. I have used my education and experience working in the electrical engineering and computer science fields, and my understanding of the knowledge,

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

creativity, and experience of a person having ordinary skill in the art in forming the opinions expressed in this declaration, as well as any other materials discussed herein.

SCOPE OF TESTIMONY

20. I understand that this IPR involves a number of subjects related to the patentability of certain claims of the '768 patent. For example, I understand that the ultimate question of whether the '768 patent would have been obvious in view of the prior art at the relevant time is at issue in this IPR. However, I understand that my testimony in this IPR is limited in scope, and that I have been asked to testify regarding a limited set of subjects relevant to this IPR. The specific topics that I have been asked to address are listed in the lettered, bold headings of the section of this declaration labeled "SUBSTANTIVE TESTIMONY." I have spent a significant amount of time, over multiple days, to analyze and express opinions about the specific topics addressed in this declaration.

21. With respect to this IPR, I have not analyzed nor expressed any opinion, nor endeavored to analyze or express an opinion, about any subject that is not expressly included in the section labeled "SUBSTANTIVE TESTIMONY." While I could, if given sufficient time, analyze and express an opinion concerning additional subjects related to the field of the invention that are not expressly included in this declaration, doing so would require a significant investment of time, over

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

multiple days, similar to the significant amount of time I devoted to the issues addressed in this declaration.

RELEVANT LEGAL STANDARDS

22. I am an electrical engineer and computer scientist by training and profession. The opinions I am expressing in this declaration involve the application of my knowledge and experience to the evaluation of the '768 patent and certain prior art with respect to the '768 patent. My knowledge of patent law is that of a lay person, albeit one who has had some experience relevant to patent law. Therefore, the attorneys from Knobbe, Martens, Olson & Bear, LLP, who represent ACS, have provided me with guidance as to the applicable patent law in this matter. The paragraphs below express my understanding of how I must apply current principles related to patentability to my analysis.

23. I understand that, in assessing the patentability of a patent claim, the Patent Office generally construes claim terms by giving them their ordinary and customary meaning, as they would have been understood by a person of ordinary skill in the art ("POSITA") at the time of the invention in view of the intrinsic record (patent specification and file history). However, I understand that the inventors may, in the patent specification, expressly define a claim term to have a meaning that differs from the term's ordinary and customary meaning. I also understand that the inventors may disavow or disclaim certain claim scope, thereby departing from the

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

ordinary and customary meaning, when the intrinsic record demonstrates that a clear and unambiguous disavowal or disclaimer has occurred. I understand that extrinsic evidence, such as relevant technical literature and dictionaries, may be useful in ascertaining how a POSITA would have understood a claim term, but the intrinsic record is the primary source for determining the meaning of claim terms. For the purposes of this review, and to the extent necessary, I have interpreted each claim term in accordance with the principles set forth in this paragraph.

24. It is my understanding that a claim is unpatentable as “anticipated” under 35 U.S.C. § 102 if a single prior art reference discloses every limitation of the claim, arranged as in the claim. I understand that a prior art reference does not anticipate a claim, however, when it discloses multiple, distinct teachings that a person of ordinary skill in the art might somehow combine to achieve the claimed invention. I understand that anticipation has not been alleged, and, thus, is not at issue in this IPR.

25. I understand that a claim is unpatentable under 35 U.S.C. § 103 if the claimed subject matter as a whole would have been obvious to a person of ordinary skill in the art at the time of the alleged invention. I also understand that an obviousness analysis takes into account the following factors, which are sometimes referred to as the *Graham* factors: (1) the scope and content of the prior art, (2) the differences between the claimed subject matter and the prior art, (3) the level of

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

ordinary skill in the art at the time of the invention, and (4) “objective indicia of non-obviousness,” also referred to as secondary considerations of non-obviousness. Those objective indicia include considerations such as whether a product covered by the claims is commercially successful due to the merits of the claimed invention, whether there was a long felt need for the solution provided by the claimed invention, whether others failed to find the solution provided by the claimed invention, whether others copied the claimed invention, and whether there was acceptance by others of the claimed invention as shown by praise from others in the field.

26. In determining the scope and content of the prior art, it is my understanding that a reference is considered appropriate prior art if it falls within the field of the inventor’s endeavor. In addition, a reference is appropriate prior art if it is reasonably pertinent to the particular problem with which the inventor was involved. A reference is reasonably pertinent if it logically would have commended itself to an inventor’s attention in considering his problem. If a reference relates to the same problem as the claimed invention, that supports use of the reference as prior art in an obviousness analysis.

27. To assess the differences between prior art and the claimed subject matter, it is my understanding that 35 U.S.C. § 103 requires the claimed invention to be considered as a whole. This “as a whole” assessment requires showing that one of ordinary skill in the art at the time of invention, confronted by the same

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

problems as the inventor and with no knowledge of the claimed invention, would have selected the elements from the prior art and combined them in the claimed manner.

28. It is my further understanding that the Supreme Court has recognized several rationales for combining references or modifying a reference to show obviousness of claimed subject matter. Some of these rationales include: combining prior art elements according to known methods to yield predictable results; simple substitution of one known element for another to obtain predictable results; a predictable use of prior art elements according to their established functions; applying a known technique to a known device (method or product) ready for improvement to yield predictable results; choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success; and some teaching, suggestion, or motivation that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention.

29. I understand that an assessment of what a reference discloses or teaches—for purposes of an anticipation analysis or an obviousness analysis—must be conducted from the perspective of a POSITA at the time of the invention. In other words, a reference discloses or teaches a claim limitation if a POSITA would, at the relevant time, interpret the reference as expressly, implicitly, or inherently

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

disclosing the claim limitation. I further understand that a reference does not need to use the exact language of the claim to disclose a claim limitation.

30. I understand that a patent owner may attempt to establish a date of invention that is earlier than the effective filing date of the patent. However, I understand that ACS is not attempting to establish an earlier invention date for the purpose of this IPR, and, thus, for the purpose of this IPR, the time of the invention (also called the “relevant time” herein) is the effective filing date of the ’768 patent. I understand that the earliest possible effective filing date of the ’768 patent is June 13, 2006 and that Petitioner has not alleged that the ’768 patent is not entitled to the June 13, 2006 filing date. Accordingly, I have assumed that June 13, 2006 is the time of invention or relevant time for purposes of this IPR.

SUBSTANTIVE TESTIMONY

A. The field of the invention and the level of ordinary skill in the art.

31. I was asked by counsel to assess the field of the invention of the ’768 patent and the level of ordinary skill in the art. In making this assessment, I have considered the ’768 Patent, my experience, and the testimony of Petitioner’s expert, Dr. Tufo.

32. In my opinion, the field of the invention of the ’768 patent is “cluster computing.” This is supported by the “Field of Disclosure” section of the patent, the specification generally, and the claims. The “Field of Disclosure” section states:

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

“The present disclosure relates to the field of cluster computer generally” The specification of the ’768 patent consistently discloses aspects of a computer cluster or methods of operating a computer cluster. And every challenged claim expressly recites a “computer cluster.”

33. Dr. Tufo did not expressly specify the field of the invention of the ’768 patent. However, Dr. Tufo refers generally to “parallel and/or distributed computing.” Ex. 1105 ¶ 40; *see also id.* ¶¶ 22-36. In my view, both “parallel computing” and “distributed computing” are different from the “cluster computing” field identified and claimed by the ’768 patent. A POSITA would understand that “parallel computing” is a category that includes “cluster computing” but “distributed computing” is different from both “cluster computing” and “parallel computing.”

34. Accordingly, I disagree with Dr. Tufo’s opinion that a POSITA “would have had a Bachelor’s degree in computer science, electrical engineering, or an equivalent field, and two years of academic or industry experience in *parallel and/or distributed computing*.” Ex. 1105 ¶ 40 (emphasis added). In my opinion, a POSITA would have had a Bachelor’s degree in computer science, electrical engineering, or an equivalent field, and two years of academic or industry experience in *cluster computing*. In conducting my analysis set forth in this Declaration, I applied my identification of the level of ordinary skill in the art, rather than Dr. Tufo’s.

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

However, my conclusions would not change even assuming that Dr. Tufo's identification of the level of ordinary skill in the art were correct.

35. At the time of the invention and presently I did and do have more training, expertise, and knowledge than a POSITA. However, I understand what a POSITA would have known and understood at the relevant time and my testimony herein is from the perspective of a POSITA at the relevant time.

B. Fundamental features of the claimed invention.

36. The '768 patent describes and claims fundamental improvements over previous efforts to allow multiple nodes to work together to perform computations in parallel. Claim 26, for example, recites a precise order of operations involving three specific nodes: (1) the third node receives, from the second node, a result of a calculation that was performed by the second node in a different claim limitation; (2) the third node performs a different calculation based on the received result; and (3) the third node sends the new result to the first node. Claim 35 recites a mechanism to allow the nodes of a computer cluster to communicate tasks and data with one another in a peer-to-peer architecture. According to the background section of the patent, one previous attempt to allow multiple nodes to work together to perform computations was "a form of grid computing known as 'distributed computing.'" Ex. 1101 at 1:44-62. The form of grid computing described in the patent relied on a "master node that manages a plurality of slave nodes or

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

computational nodes.” *Id.* at 1:51-53. The “master kernel . . . handles all input, output, and scheduling of the other kernels (the computational kernels or slave kernels). Computational kernels receive commands and data only from the node running the master kernel.” *Id.* at 1:55-59. The nodes “generally do not communicate with one another as peers.” *Id.* at 1:46-47. The patent distinguishes this form of grid computing from a “computer cluster having peer-to-peer node architecture.” *Id.* at 12:33-40.

37. The ’768 patent expressly claims the “peer-to-peer node architecture” disclosed in the specification. Specifically, claim 35 recites a computer cluster with multiple nodes and “a mechanism to communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture.” Claim 35 further recites that “the computer cluster node is configured to . . . communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other.”

38. The specification discloses a particular mechanism for the nodes to communicate using a “peer-to-peer architecture.” Specifically, multiple “cluster node modules” establish connections from each cluster node module to every other cluster node module and exchange messages in a process that “provides the peer-to-peer behavior of the cluster node modules.” Ex. 1101 at 23:51-52, 24:39-40,

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

24:52-53, 25:1-2, 25:9-10, 25:23-24, 25:37-38. Dependent claim 27 recites a “cluster node module.”

39. A POSITA would understand, in view of the specification, that “peer-to-peer behavior” is an essential feature of both the “peer-to-peer architecture” limitation of claim 35 and the narrower “cluster node module” limitation of claim 27. The “peer-to-peer behavior” includes at least that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node. Accordingly, consistent with the inventors’ fundamental architecture of the invention, the Board should construe the “peer-to-peer architecture” limitation of claim 35 to require at least “an architecture in which each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.” Similarly, the Board should construe the slightly narrower “cluster node module” limitation of claim 27 to mean “a module that cooperates with other cluster node modules to establish intercommunication among nodes in a computer cluster and to exchange messages such that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.”

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

C. Claim construction of “a mechanism to communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture” (claim 35)

40. Claim 35 recites “*a mechanism to communicate* results of evaluation with other computer cluster nodes *using a peer-to-peer architecture*” and “wherein the computer cluster node is configured to . . . communicate at least some of the user instructions using *the mechanism for the nodes to communicate with each other*” (emphases added). With respect to these two limitations, there are two claim construction issues relevant to the Board’s decision whether to institute an IPR: (1) determining that both limitations refer to the same “mechanism to communicate . . . using a peer-to-peer architecture” and (2) determining the meaning of “peer-to-peer architecture.”

1. The “wherein” clause refers back to the same “mechanism to communicate . . . using a peer-to-peer architecture” introduced earlier in the claim.

41. A POSITA would understand that the “wherein” clause uses the standard claim-drafting technique of using a shorthand phrase introduced by the definite article “the”—specifically, “the mechanism for the nodes to communicate with each other”—to refer to a limitation recited earlier in the claim. A POSITA would also understand that the *only* earlier limitation of claim 1 that recites a “mechanism” that provides antecedent basis for “the mechanism” of the “wherein” clause, is “a mechanism to communicate . . . using a peer-to-peer architecture.”

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

Accordingly, the Board should determine that the “wherein” clause refers back to the same “mechanism to communicate . . . using a peer-to-peer architecture” recited earlier in the claim. This means that “results of evaluation” and “at least some of the user instructions” must be communicated using the claimed “mechanism to communicate . . . using a peer-to-peer architecture.”

2. “peer-to-peer architecture”

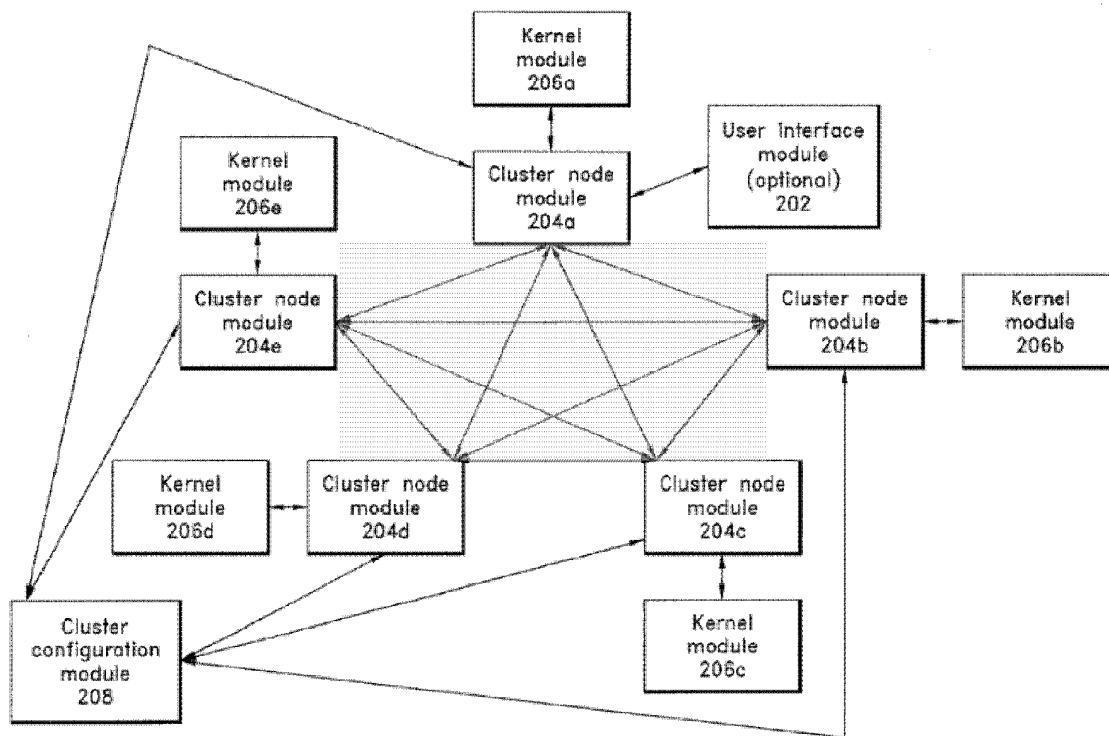
42. The claimed communication mechanism must use a “peer-to-peer architecture.” A POSITA would understand that, in the context of cluster computing, the adjective “peer-to-peer” ordinarily means that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node. A POSITA would also look to the specification of the ’768 patent to confirm this understanding.

43. As explained below, a POSITA would conclude from the specification that communicating tasks and data with other nodes without the tasks and data being required to go through a central server or master node is a primary function of the “peer-to-peer architecture” of the ’768 patent. Therefore, “peer-to-peer architecture,” in the context of the ’768 patent, should be construed to require at least “an architecture in which each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.”

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

44. A POSITA would understand that the “peer-to-peer architecture” required by claim 35 is one feature of the interconnected “cluster node modules” disclosed by the ’768 patent and recited by claim 27. Accordingly, the ’768 patent’s disclosure of the “peer-to-peer architecture” enabled by the “cluster node modules” is instructive of what “peer-to-peer architecture” means in the context of the ’768 patent. Figure 2 illustrates that each cluster node module is connected to every other cluster node module, as shown by the yellow highlighting in the annotated figure below.

*FIG. 2*

Ex. 1101, Fig. 2 (highlighting added). The specification confirms that “each cluster node module 204a-e is connected to all other cluster node modules” and “[t]he

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

cluster node modules 204a-e establish communication with one another” through “direct connections” between each cluster node module. Ex. 1101 at 23:13-14, 23:51-56.

45. As illustrated and described, the cluster node modules provide a direct connection from each node to every other node. In addition, the specification discloses that a process for passing messages among the cluster node modules “provides the peer-to-peer behavior of the cluster node modules,” allowing them to “interact on a pair-wise or collective basis.” Ex. 1101 at 25:22-38.

46. In view of the specification, a POSITA would understand that the message-passing process allows each cluster node module to communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node. Specifically, the specification discloses that “[c]ommunications can occur between any two or more cluster node modules” and that “[e]ach of the cluster node modules 204a-e is in communication with respective kernel modules 206a-e,” thereby enabling “MPI calls and advanced cluster commands” to be “used to parallelize program code received from an optional user interface module 208 and *distribute tasks* among the kernel modules 206a-e.” Ex. 1101 at 6:9-25 (emphasis added).

47. In addition, the specification distinguishes a peer-to-peer architecture from the master-slave architecture typically used for “a form of grid computing

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

known as ‘distributed computing.’” Ex. 1101 at 1:44-62; *see also id.* at 12:30-33 (gridMathematica connects kernels “in a master-slave relationship rather than a peer-to-peer relationship”). The specification describes the following characteristics of the master-slave architecture of grid computing:

Grid computers include at least one node known as a master node that manages a plurality of slave nodes or computational nodes. In gridMathematica, each of a plurality of kernels runs on a single node. ***One kernel is designated the master kernel, which handles all input, output, and scheduling of the other kernels (the computational kernels or slave kernels). Computational kernels receive commands and data only from the node running the master kernel.***

Id. at 1:51-59 (emphasis added). By distinguishing the peer-to-peer architecture of the invention from the master-slave architecture, the specification implicitly indicates that the peer-to-peer architecture does not have the identified characteristics of the master-slave architecture. Accordingly, by contrast to the master-slave architecture, each node in the disclosed peer-to-peer architecture is able to handle “scheduling” and communicating (including distributing, sending, and receiving) “commands and data” to and from other nodes without requiring them to go through a central server or master node. A POSITA would understand that the communication of tasks and data falls within these “scheduling” and communicating “commands and data” functions.

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

48. Accordingly, a POSITA would understand that the communication of tasks and data is a primary function that the nodes of the disclosed peer-to-peer architecture of the '768 patent can handle without requiring a central server or master node. The Board should construe "peer-to-peer architecture," in the context of the '768 patent, to require at least "an architecture in which each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node."

D. Claim construction of "cluster node module" (claim 27)

49. Claim 27 recites "wherein the single-node kernel is configured to send the first packet to a local cluster node module." The phrase "cluster node module" was not a common or well-known technical phrase having any specific meaning in the relevant art at the time of the invention. Indeed, even today, more than 14 years after the earliest effective filing date of the '768 patent, a search for patents or patent application publications that use the phrase "cluster node module" returns only patents and applications filed by the inventors of the '768 patent. Accordingly, there was no ordinary meaning of "cluster node module" in the relevant field at the relevant time. Further, while the individual terms "cluster," "node," and "module," were known in the relevant field, a POSITA would understand that no ordinary meaning for the combined phrase "cluster node module" could reliably be composed by attempting to combine the meanings of the individual terms.

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

50. In view of the claim language and specification, a POSITA would understand that the inventors coined the phrase “cluster node module” to encapsulate essential features of the modules that interconnect the nodes in a preferred embodiment of the invention.¹ I understand that, because “cluster node module” is a coined phrase, reliance on the specification is necessary to ascertain its meaning. As explained below, in view of the specification, the Board should construe “cluster node module” to mean “a module that cooperates with other cluster node modules to establish intercommunication among nodes in a computer cluster and to exchange messages such that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.”

51. The specification discloses several *optional* components of the cluster node modules. For example:

FIG. 3 shows one embodiment of a cluster node module 204 implementing MPI calls and advanced MPI functions. ***In the embodiment shown in FIG. 3***, cluster node module 204 includes MPI module 202, advanced functions module 304, received message queue 306, and message receiving queue 308.

¹ Several claims of the '768 patent, including claim 1, recite “a plurality of nodes” but not “cluster node modules.”

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

Ex. 1101 at 12:41-46 (emphases added). Because the components depicted in Figure 3 are part of “one embodiment,” a POSITA would understand that they are optional components of the cluster node modules. Dependent claims of U.S. Patent No. 8,082,289 (which is related to the ’768 patent) specifically reciting that the cluster node modules comprise an MPI module, advanced functions module, received message queue, and message receiving queue also demonstrate that these components are optional parts of the cluster node modules. IPR2020-01608, Ex. 1001, claim 8 (received message queue); claim 9 (message receiving queue); claim 13 (advanced functions module); claim 26 (MPI module). Accordingly, a POSITA would understand that the basic “cluster node modules” do not require an MPI module, advanced functions module, received message queue, and message receiving queue.

52. Rather than interpreting “cluster node module” to include optional components such as those depicted by Figure 3, a POSITA would look to the specification to limit the “cluster node module” to its essential features. The specification unambiguously discloses that cluster node modules are configured to establish intercommunication with one another, communicate messages among themselves and with kernels, and, through such message-passing, “provide[] the peer-to-peer behavior of the cluster node modules.” Ex. 1101 at 23:51-52, 24:39-40, 24:52-53, 25:1-2, 25:9-10, 25:23-24, 25:37-38. Significantly, the specification does

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

not say these disclosed features apply just to “one embodiment” of the cluster node modules or otherwise suggest they are optional. Therefore, a POSITA would understand that the capability to establish intercommunication among all cluster node modules and to exchange messages to provide “peer-to-peer behavior” are essential features of the cluster node modules.

53. A POSITA would further examine both the ordinary meaning of “peer-to-peer” and the specification to determine what is meant by the cluster node modules’ “peer-to-peer behavior.” As explained above with respect to the “peer-to-peer architecture” limitation, a POSITA would conclude that “peer-to-peer behavior” includes at least that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.

54. Accordingly, the Board should construe “cluster node module” to mean “a module that cooperates with other cluster node modules to establish intercommunication among nodes in a computer cluster and to exchange messages such that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.”

E. The Petition does not establish claim 26 is unpatentable.

55. The Petition does not show that the prior art discloses “wherein the third node comprises a third hardware processor with a plurality of processing cores,

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node.”

56. This limitation recites a precise order of operations involving three specific nodes: (1) the third node receives, from the second node, a result of a calculation that was performed by the second node in a different claim limitation; (2) the third node performs a different calculation based on the received result; and (3) the third node sends the new result to the first node. The Petition relies on Schreiner 1 and Schreiner 3 for this limitation. Pet. at 47-50.

57. The Petition fails to prove its allegation that the prior art discloses this detailed limitation. To prove that allegation, the Petition would need to show where the prior art teaches the precise order of operations, involving three specific nodes, specified by the claim. The Petition does not do that. Instead, the Petition refers to the prior art’s general disclosure that multiple nodes may be involved in evaluating mathematical expressions and then speculates that the prior art could work in a manner that would meet the claim limitation. Pet. at 44-47.

58. The Petition first relies on “the example of Figure 5” of Schreiner 1. *Id.* at 48. It is clear on the face of Petitioner’s argument that Figure 5 does not actually disclose every detail about the claim limitation and that Petitioner is merely

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

speculating about how Figure 5 allegedly could work. The Petition asserts that a “POSITA would understand that the third node then *could* execute a second mathematical expression using that result, and in fact, that is *likely why* the third node requested the result in the first place.” *Id.* at 48-49 (emphases added). The Petition further speculates that “the third node *could* send the result of the second mathematical expression to the first (root) node.” *Id.* at 49 (emphasis added).

59. The Petition also relies on an “example . . . illustrated in Schreiner 3 Fig. 1.” *Id.* at 49. But the cited example from Schreiner 3 is about a mechanism for “the *logging* of task return values” (Ex. 1110 at 1) and neither Schreiner 3 nor the Petition explains how Figure 1 relates to the evaluation of mathematical expressions by multiple nodes. The Petition also makes clear that Figure 1 does not actually disclose every detail about the claim limitation and that Petitioner is merely speculating about how Figure 1 allegedly could work. The Petition speculates that “once [client 1] receives the result message from node ‘client n,’ it *may* use it to perform at least a second mathematical expression evaluation.” Pet. at 49-50. Petitioner offers no evidence or explanation supporting this speculation. *Id.*

60. Because the Petition merely speculates about what prior art nodes “could” or “may” do, the Petition fails to prove Petitioner’s assertion that the prior art discloses the claim limitation.

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

F. The Petition does not establish claim 27 is unpatentable.

61. Claim 27 depends from claim 26 and, thus, is patentable for at least the same reasons that claim 26 is patentable. In addition, the prior art does not disclose the “cluster node module” limitation recited by claim 27.

62. As explained above, because “cluster node module” is a coined phrase encapsulating the essential features of the cluster node modules disclosed in the specification, the phrase should be construed to mean “a module that cooperates with other cluster node modules to establish intercommunication among nodes in a computer cluster and to exchange messages such that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.” These essential features of the cluster node modules are fundamental parts of the claimed invention of claim 27.

63. The Petition alleges that the kernel’s “local scheduler” of the prior art is a “cluster node module.” Pet. at 55. However, the Petition does not allege or prove that the scheduler allows each node to “communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.” *Id.* In fact, Schreiner 1 expressly discloses that at least distributing tasks requires a master node: “All remote schedulers send new tasks to the root node scheduler which distributes them among all machines.” Ex. 1108 at 316. Schreiner

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

1 further confirms “the root is in charge of task scheduling” and “the root sees every task created in the session.” *Id.* at 323. A POSITA would understand that the “root node” of Schreiner 1 is a “master node.” Accordingly, because the purported cluster node module of the prior art does not “communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node,” it does not satisfy the “cluster node module” limitation as properly construed.

64. I understand that the Petition does not allege that a POSITA would modify Schreiner 1 or any of the other Distributed Maple references to not require the root node to be involved in the distribution of tasks. Accordingly, I understand it is not necessary to analyze whether it would have been obvious to make such a modification to assess the Petition’s arguments. Nevertheless, I further analyzed the Distributed Maple references and concluded that a POSITA would understand that Schreiner 1’s disclosure that tasks must be distributed through the root node is a fundamental feature of Distributed Maple that a POSITA would not have modified, at least because that feature enables the fault tolerance mechanism disclosed in Schreiner 1. As part of its fault tolerance mechanism, Schreiner 1 discloses a logging mechanism in which “the root sees every result computed in the session” and that “the root sees every task created in the session.” Ex. 1008 at 323. Schreiner 1 further discloses that “the delegation of all logging activities to a single root node that also performs the task scheduling decisions” are important features that enable the fault

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

tolerance mechanism to be “simple” and to avoid “global snapshots as required in message passing programs” and that this logging mechanism has “runtime overhead” that is “very moderate” and the fault tolerance mechanism does not require much extra overhead.” *Id.* A POSITA would understand, in view of the disclosure of Schreiner 1, that distributing tasks through the root node is required to take advantage of the disclosed fault tolerance mechanism. Therefore, even if fault tolerance were optional at the user level—that is, some users could turn on the provision of fault tolerance while others could turn it off a POSITA would not modify Distributed Maple to not require distributing tasks through the root node, because such a modification would destroy the ability of users to use the fault tolerance mechanism of Distributed Maple.

G. The Petition does not establish claim 29 is unpatentable.

65. Claim 29 includes the same limitations of claim 26 analyzed above. The Petition acknowledges that “[c]laim 29 is identical to claim 26” except for two small differences that do not involve the claim 26 limitation analyzed above. Pet. at 56. Accordingly, claim 29 is patentable for at least the same reasons claim 26 is patentable.

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

H. The Petition does not show that the prior art discloses the “peer-to-peer architecture” limitation (claim 35).

66. Claim 35 recites “a mechanism to communicate results of evaluation with other computer cluster nodes using a *peer-to-peer architecture*” (emphasis added). As explained above, the Board should construe “peer-to-peer architecture” to require at least “an architecture in which each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.” The alleged prior art does not disclose the “peer-to-peer architecture” limitation at least because it expressly requires at least all tasks to go through a central server or master node.

67. Specifically, Schreiner 1 discloses: “All remote schedulers send new tasks to the root node scheduler which distributes them among all machines.” Ex. 1108 at 316. It further discloses that “[s]ince the root is in charge of task scheduling, the root sees every task created in the session.” *Id.* at 323. A POSITA would understand that the “root node” of Schreiner 1 is a “master node.” Accordingly, the prior art’s requiring at least tasks to go through a master node means that the prior art does not disclose a “peer-to-peer architecture.”

68. The Petition relies on Schreiner 1’s disclosure that a node needing “to send a message to one of its peers . . . can thus establish a direct connection for message transfers” and argues that Figure 1 of Schreiner 1 depicts “direct, peer-to-

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

peer communications.” Pet. at 21. But these disclosures do not negate Schreiner 1’s express disclosure that its architecture requires all tasks to go through a master node. A POSITA would understand that the creation of a “peer connection” does not make Schreiner’s architecture “peer-to-peer” when Schreiner 1 expressly disallows communication of tasks using a peer-to-peer architecture by requiring all tasks to go through a master node.

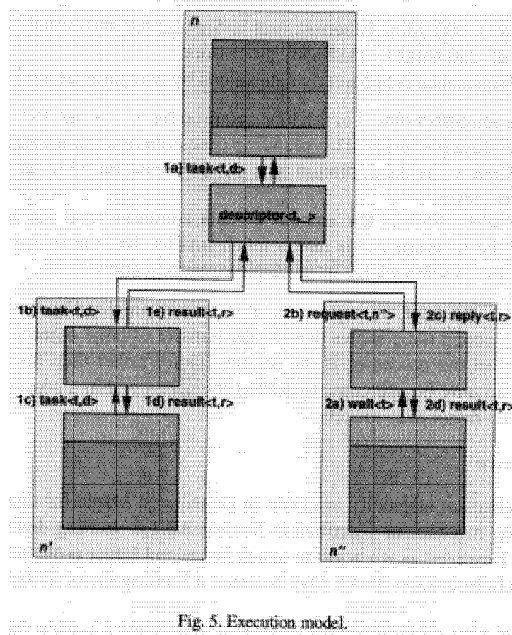
69. Nothing in the prior art even suggests that tasks can be communicated using direct node-to-node communication without requiring the tasks to go through the “single root node.” Indeed, Schreiner 1’s express disclosure that “[a]ll remote schedulers send new tasks to the root node scheduler which distributes them among all machines” (Ex. 1108 at 316) and “the root is in charge of task scheduling” (*id.* at 323) undermines any possible contention that the disclosed architecture enables direct node-to-node communication of tasks without the tasks being required to go through the “single root node.”

70. Petitioner’s arguments about the communication of result messages fail to show that the prior art satisfies the “peer-to-peer architecture” limitation at least because they do not negate the prior art’s express teaching that tasks are required to go through a master node. Further, the Petition even fails to show that the prior art communicates result messages without requiring the involvement of a master node. Petitioner attempts to imply there is a link between Schreiner 1’s sentence, on page

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

319, about an “idle kernel” returning “a message result” with the sentence on page 311 that “additional socket connections between remote scheduler instances are created on demand” by quoting those sentences directly adjacent to each other in the Petition. Pet. at 29. But those sentences are neither adjacent nor logically linked in Schreiner 1. The sentences are eight pages apart, in different sections, and Schreiner 1 does not suggest they are related to each other. The page 311 quote is in the “Software architecture” section and relates to the “software architecture” of Figure 1. Ex. 1108 at 311. The page 319 quote, by contrast, is in the “System and execution model” subsection of the “Fault tolerance” section and relates to the “execution model” of Figure 5. *Id.* at 319. Significantly, the “execution model depicted in Fig. 5,” as shown below, does **not** depict any direct node-to-node communications that do not require data to go through the root node:



IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

Id. at 320. Indeed, *every* connection depicted on Figure 5 includes the root node.

71. The Petition also relies on Figure 1 of Schreiner 3, arguing that the “6 result message” box depicts “client n” sending “client 1” a “result message.” Pet. at 30. However, the Petition ignores the “7 store message” box, which shows that “client n” sends the same “result” to the “root.” Ex. 1110 at 5 (“When the result is computed the root receives it in a *store message* and saves it to the hard disk (see Figure 1).”); 9 (“If a node has computed the result of a task, it sends it in a *result message* to the node which created this task and also sends a duplicate in a *store message* to the root.”). Accordingly, Schreiner 3 sends results to a master node. Further, Schreiner 3’s communication of results does not overcome the prior art’s express disclosure that tasks are required to go through a master node.

72. Therefore, the Petition does not prove that the prior art discloses a “peer-to-peer architecture” in which “each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.”

73. I understand that the Petition does not allege that a POSITA would modify Schreiner 1 or any of the other Distributed Maple references to not require the root node to be involved in the distribution of tasks. Accordingly, I understand it is not necessary to analyze whether it would have been obvious to make such a modification to assess the Petition’s arguments. Nevertheless, I further analyzed the

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

Distributed Maple references and concluded that a POSITA would understand that Schreiner 1's disclosure that tasks must be distributed through the root node is a fundamental feature of Distributed Maple that a POSITA would not have modified, at least because that feature enables the fault tolerance mechanism disclosed in Schreiner 1. As part of its fault tolerance mechanism, Schreiner 1 discloses a logging mechanism in which "the root sees every result computed in the session" and that "the root sees every task created in the session." Ex. 1108 at 323. Schreiner 1 further discloses that "the delegation of all logging activities to a single root node that also performs the task scheduling decisions" are important features that enable the fault tolerance mechanism to be "simple," to avoid "global snapshots as required in message passing programs" and to have "runtime overhead" that is "very moderate." *Id.* at 323-324. A POSITA would understand, in view of the disclosure of Schreiner 1, that distributing tasks through the root node is required to take advantage of the disclosed fault tolerance mechanism. Therefore, even if fault tolerance were optional at the user level—that is, some users could turn on the provision of fault tolerance while others could turn it off—a POSITA would not modify Distributed Maple to not require distributing tasks through the root node, because such a modification would destroy the ability of users to use the fault tolerance mechanism of Distributed Maple.

IPR2021-00020

*NVIDIA Corp. v. Advanced Cluster Sys.***I. The Petition does not establish the prior art discloses communication of “user instructions” using the “peer-to-peer architecture.”**

74. As explained above, the “wherein” clause of claim 35 requires the same “peer-to-peer architecture” used to communicate “results” to also communicate “at least some of the user instructions.” The Petition does not show that the prior art meets this limitation. The Petition relies on the prior art’s communication of *tasks* as allegedly meeting the limitation, but as explained above, the prior art expressly discloses that its communication of tasks relies on the tasks being required to go through a root node, meaning that it does not use a peer-to-peer architecture.

75. The Petition specifically alleges that, in the prior art, “the root node accepts . . . Distributed Maple instructions and communicates these instructions using the node-to-node messaging mechanism.” Pet. at 51 (citing Ex. 1108 (Schreiner 1) at 309-314). But Schreiner 1 never says that those instructions are communicated using a “peer-to-peer architecture” in which the instructions are not required to go through a central server or master node. Ex. 1108 at 309-314.

76. In fact, the cited pages of Schreiner 1 indicate that instructions are sent as *tasks*. *Id.* For example, the Petition acknowledges that “the user’s instruction dist[start] is communicated to other nodes as *tasks*.” Pet. at 51 (emphasis added). The Petition further asserts that dist.maple source code confirms that these instructions are communicated as *task* messages. *Id.* at 51-52. However, as

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

explained above, Schreiner 1 expressly discloses that tasks, even if viewed as task messages, are distributed by the root node: “All remote schedulers send new tasks to the root node scheduler which distributes them among all machines.” Ex. 1108 at 316; *see also id.* at 323 (“the root is in charge of task scheduling”). Accordingly, because the communication of all tasks must go through a master node, the prior art does not disclose using a “peer-to-peer architecture” for such communication.

77. I understand that the Petition does not allege that a POSITA would modify Schreiner 1 or any of the other Distributed Maple references to not require the root node to be involved in the distribution of tasks. Accordingly, I understand it is not necessary to analyze whether it would have been obvious to make such a modification to assess the Petition’s arguments. Nevertheless, I further analyzed the Distributed Maple references and concluded that a POSITA would understand that Schreiner 1’s disclosure that tasks must be distributed through the root node is a fundamental feature of Distributed Maple that a POSITA would not have modified, at least because that feature enables the fault tolerance mechanism disclosed in Schreiner 1. As part of its fault tolerance mechanism, Schreiner 1 discloses a logging mechanism in which “the root sees every result computed in the session” and that “the root sees every task created in the session.” Ex. 1108 at 323. Schreiner 1 further discloses that “the delegation of all logging activities to a single root node that also performs the task scheduling decisions” are important features that enable the fault

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

tolerance mechanism to be “simple,” to avoid “global snapshots as required in message passing programs” and to have “runtime overhead” that is “very moderate.”

Id. at 323-324. A POSITA would understand, in view of the disclosure of Schreiner 1, that distributing tasks through the root node is required to take advantage of the disclosed fault tolerance mechanism. Therefore, even if fault tolerance were optional at the user level—that is, some users could turn on the provision of fault tolerance while others could turn it off—a POSITA would not modify Distributed Maple to not require distributing tasks through the root node, because such a modification would destroy the ability of users to use the fault tolerance mechanism of Distributed Maple.

J. Patent Owner’s SEMTM and SETTM products embody the challenged claims of the ’768 patent.

78. I have also been asked by Patent Owner to provide my opinion on whether Patent Owner’s SEMTM and SETTM products practice the challenged claims of the ’768 patent as they relate to objective indicia of non-obviousness. Based on my review of technical documents which I understand were created near the time of the invention and accurately describe the SEMTM and SETTM products, and limited information about the SEMTM and SETTM products provided by Dr. Dean Dager, one of the primary developers of the SEMTM and SETTM products, it is my opinion

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

that the SEMTM and SETTM products practice all limitations of at least claims 26, 29, and 35 of the '768 patent.

79. In addition to the '768 patent, I reviewed the following documents in arriving at my conclusion that the SEMTM and SETTM products embody claims 26, 29, and 35 of the '768 patent:

- SEMTM Poster (Ex. 2009);
- SEMTM PingPong Benchmark (Ex. 2010);
- SEMTM Manual (Ex. 2011);
- SEMTM White Paper (Ex. 2012);
- SETTM White Paper (Ex. 2013);
- SETTM Manual (Ex. 2014);
- SETTM Presentation (Ex. 2015);
- SETTM Datasheet (Ex. 2016).

80. Exhibit 2031 is a claim chart that accurately sets forth my analysis showing that the SEMTM product embodies at least claims 26, 29, and 35 of the '768 patent. I adopt Exhibit 2031 as part of my testimony in this Declaration.

81. Exhibit 2032 is a claim chart that accurately sets forth my analysis showing that the SETTM product embodies at least claims 26, 29, 35 of the '768 patent. I adopt Exhibit 2032 as part of my testimony in this Declaration.

IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

82. As identified in the claim charts, with respect to some aspects of certain claim limitations, I also considered limited information about the SEMTM and SETTM products provided by Dr. Dager. In my view, the information provided by Dr. Dager confirms that the SEMTM and SETTM products include architectural and functional features that are at least implied or suggested by the technical documents and that I would expect the SEMTM and SETTM products to have based on my review of the technical documents. Further, the technical documents are entirely consistent with, and fully support, the information provided by Dr. Dager.

83. As shown in Exhibits 2031 and 2032, the SEMTM and SETTM products are software products that, when installed on a computer cluster, result in a computer cluster having the architectural and functional features included in claims 26, 29 and 35, including the “mechanism for the nodes to communicate . . . using asynchronous calls” of claims 26 and 29 and the “peer-to-peer architecture” of claim 35, which enables each node to “communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.” As further explained in Exhibits 2031 and 2032, the SEMTM and SETTM products also practice the other limitations set forth in at least claims 26, 29, and 35.

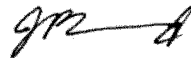
IPR2021-00020

NVIDIA Corp. v. Advanced Cluster Sys.

DECLARATION

84. I declare that all statements made herein on my own knowledge are true and that all statements made on information and belief are believed to be true, and further, that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Dated: 2/9/2021



Jaswinder Pal Singh, Ph.D

EXHIBIT 7

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

NVIDIA CORPORATION,
Petitioners,

v.

ADVANCED CLUSTER SYSTEMS, INC.
Patent Owner

Case No. IPR2021-00019
U.S. Patent 10,333,768

DECLARATION OF JASWINDER PAL SINGH, Ph.D.

Nvidia Corp. v. Advanced Cluster Systems IPR2021-00019 Advanced Cluster Systems Ex. 2001

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

1. I, Jaswinder Pal Singh, Ph.D., am making this declaration at the request of Patent Owner Advanced Cluster Systems, Inc. (“ACS”) in the matter of the *Inter Partes* Review No. IPR2021-00019 of U.S. Patent No. 10,333,768 (“the ’768 patent”). I understand that this declaration is being submitted in the IPR as Exhibit 2001.

2. I am being compensated for my work in this matter at my standard hourly rate for consulting services. My compensation in no way depends on the outcome of this proceeding.

3. In conducting the analysis and forming the opinions set forth in this Declaration, I considered, in addition to my knowledge and expertise in the relevant field, at least the following written materials:

- ’768 patent (Ex. 1001);
- File history of the ’768 patent (Ex. 1002);
- NVIDIA’s Petition (Paper 1);
- Declaration of Dr. Henry Tufo (Ex. 1005);
- Declaration of Dr. Wolfgang Schreiner (Ex. 1006);
- Schreiner 1 (Ex. 1008);
- Schreiner 2 (Ex. 1009);
- Schreiner 3 (Ex. 1010);
- Maple Guide (Ex. 1011); and

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

- Any other document referenced herein.

PROFESSIONAL BACKGROUND

4. I am currently a Professor of Computer Science at Princeton University, a position I have held for more than 15 years. Before becoming a Full Professor, I was an Assistant, and then Associate, Professor at Princeton.

5. I received a Bachelor's Degree in Electrical Engineering and Computer Science, *summa cum laude*, from Princeton in 1987. I received an M.S. Degree in Electrical Engineering in 1989 and a Ph.D. Degree in Electrical Engineering in 1993, both from Stanford University. For my Ph.D. thesis, I researched the boundary of applications and parallel computer systems, including programming models, and the interactions between software and hardware. I understand that my current curriculum vitae (CV) is being submitted in this proceeding as Exhibit 2002.

6. I joined the faculty at Princeton in 1995, where I served as an Assistant Professor in the Computer Science Department. My initial research area included parallel computer systems and applications with a focus on (i) scalable multiprocessing in both the shared address space and message passing paradigms, including studying the programming and performance tradeoffs between the two, (ii) understanding how shared address space abstractions might be supported using commodity parts, and how applications might be restructured for such systems, and

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

(iii) the use of high-performance computing in biology, medicine, cosmology, and other areas.

7. From 1999 to 2010 I was the director of Princeton's "Program in Integrative Computer and Application Sciences" (PICASso), a multi-department, university-wide interdisciplinary program focused on scalable parallel and distributed computing at the boundary of computer science and a broad range of application sciences. Directing that program required me to develop and foster new interdisciplinary courses, new practice-oriented workshops in scalable computing based on needs gathered from multiple departments, new interdisciplinary seminar series, and programs in effective written and oral scientific communication. I oversaw the creation of courses for an interdisciplinary computational science curriculum, taught by faculty in different departments, in addition to parallel and distributed computing courses. From 2003 to the present, I have been on the Executive Committee at Princeton's Institute for Computational Science and Engineering (PICSSiE). That permanent institute came out of the PICASso program that I led, and the Institute has taken over PICASso's activities as well, including a permanent Graduate Certificate Program in Computational Science and Engineering at Princeton that I initially developed.

8. In addition to my concurrent work as a professor, beginning in 1999, I was a member of the Technical Advisory Board at NOAA Geophysical Fluid

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

Dynamics Laboratory. In that role I aided in the shift from vector to parallel computing, and in the procurement of large parallel computing and analysis infrastructures.

9. At Princeton, I have taught many courses in areas relevant to the technology disclosed in the '768 patent. These include undergraduate and graduate courses in parallel computing, parallel architecture and programming, scalable infrastructure and services, and in analysis of data. I taught several new interdisciplinary courses I created for the PICASSo program regarding parallel and distributed computing. I also created and taught courses in Marketplace Design and at the boundary of technology, business, marketplaces, and economics, including one which has been dubbed the "CTO Course."

10. In the course of my work at Princeton, I have supervised more than 15 students engaged in thesis research projects in support of Master of Science and Doctor of Philosophy degrees. Those projects have included topics such as new programming models for scalable parallel computing, peer-to-peer publish-subscribe systems, and a comparison of programming models for high-performance parallel computing.

11. I am the author of a leading textbook on parallel computing, "Parallel Computer Architecture: A Hardware-Software Approach," by David E. Culler,

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

Jaswinder Pal Singh with Anoop Gupta, Morgan Kaufmann Publishers, 1999. I have also published more than 100 refereed conference and journal publications.

12. I am a member of the Association for Computing Machinery and the Institute of Electrical and Electronics Engineers.

13. I have been recognized for my research and other professional endeavors. I was honored with a Presidential Early Career Award for Scientists and Engineers (PECASE), 1997, awarded to twenty young scientists and engineers in the United States selected from all areas of science/engineering by the National Science Foundation, and the Sloan Research Fellowship, 1998, awarded to ten Computer Scientists nationwide by the Alfred Sloan Foundation. Also, my 2005 publication for the International Distributed Event-based Systems (DEBS) was in 2019 voted as the most influential paper of DEBS 2005.

14. My professional activities further include serving on corporate boards of directors, including the companies 8x8, Inc. and Hiro, PBC (formerly Blockstack, PBC).

15. I have accepted invitations to evaluate research and education programs and to evaluate proposals, including for the US Government (NOAA's procurement programs and the National Science Foundation) and the Swedish Government (Evaluating the ARTES/PAMP national program for scientific research and graduate education in high-performance computing, as well as other programs for the Swedish

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

Research Council). I have also served as an external evaluator for faculty hiring at Uppsala University, Sweden and the University of Copenhagen, Denmark. I have delivered many invited presentations and lectures at a variety of international and domestic venues with details provided in my curriculum vitae.

16. I have served on the program committees and as program chair and track chair of many ACM, IEEE and other conferences in scalable computing, including the International Symposium on Computer Architecture, Supercomputing/SC, SIGMETRICS Conf. on Measurement and Modeling of Computer Systems, Principles and Practice of Parallel Programming, Symposium on Parallel Algorithms and Architectures, International Conference on Supercomputing, International Parallel Processing Symposium, International Conference on Parallel Processing etc., as well as several ACM and IEEE workshops on scalable computing, data analysis and mining, architecture, and languages and compilers.

17. In my education, research, and consulting work, I have used and designed many systems for parallel and clustered computation.

18. I am a named inventor of six issued patents, including U.S. Patent Nos. 7,080,073; 6,915,294; 7,415,469; 7,103,838; 10,796,276; and 10,796,277.

19. I have used my education and experience working in the electrical engineering and computer science fields, and my understanding of the knowledge,

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

creativity, and experience of a person having ordinary skill in the art in forming the opinions expressed in this declaration, as well as any other materials discussed herein.

SCOPE OF TESTIMONY

20. I understand that this IPR involves a number of subjects related to the patentability of certain claims of the '768 patent. For example, I understand that the ultimate question of whether the '768 patent would have been obvious in view of the prior art at the relevant time is at issue in this IPR. However, I understand that my testimony in this IPR is limited in scope, and that I have been asked to testify regarding a limited set of subjects relevant to this IPR. The specific topics that I have been asked to address are listed in the lettered, bold headings of the section of this declaration labeled "SUBSTANTIVE TESTIMONY." I have spent a significant amount of time, over multiple days, to analyze and express opinions about the specific topics addressed in this declaration.

21. With respect to this IPR, I have not analyzed nor expressed any opinion, nor endeavored to analyze or express an opinion, about any subject that is not expressly included in the section labeled "SUBSTANTIVE TESTIMONY." While I could, if given sufficient time, analyze and express an opinion concerning additional subjects related to the field of the invention that are not expressly included in this declaration, doing so would require a significant investment of time, over

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

multiple days, similar to the significant amount of time I devoted to the issues addressed in this declaration.

RELEVANT LEGAL STANDARDS

22. I am an electrical engineer and computer scientist by training and profession. The opinions I am expressing in this declaration involve the application of my knowledge and experience to the evaluation of the '768 patent and certain prior art with respect to the '768 patent. My knowledge of patent law is that of a lay person, albeit one who has had some experience relevant to patent law. Therefore, the attorneys from Knobbe, Martens, Olson & Bear, LLP, who represent ACS, have provided me with guidance as to the applicable patent law in this matter. The paragraphs below express my understanding of how I must apply current principles related to patentability to my analysis.

23. I understand that, in assessing the patentability of a patent claim, the Patent Office generally construes claim terms by giving them their ordinary and customary meaning, as they would have been understood by a person of ordinary skill in the art ("POSITA") at the time of the invention in view of the intrinsic record (patent specification and file history). However, I understand that the inventors may, in the patent specification, expressly define a claim term to have a meaning that differs from the term's ordinary and customary meaning. I also understand that the inventors may disavow or disclaim certain claim scope, thereby departing from the

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

ordinary and customary meaning, when the intrinsic record demonstrates that a clear and unambiguous disavowal or disclaimer has occurred. I understand that extrinsic evidence, such as relevant technical literature and dictionaries, may be useful in ascertaining how a POSITA would have understood a claim term, but the intrinsic record is the primary source for determining the meaning of claim terms. For the purposes of this review, and to the extent necessary, I have interpreted each claim term in accordance with the principles set forth in this paragraph.

24. It is my understanding that a claim is unpatentable as “anticipated” under 35 U.S.C. § 102 if a single prior art reference discloses every limitation of the claim, arranged as in the claim. I understand that a prior art reference does not anticipate a claim, however, when it discloses multiple, distinct teachings that a person of ordinary skill in the art might somehow combine to achieve the claimed invention. I understand that anticipation has not been alleged, and, thus, is not at issue in this IPR.

25. I understand that a claim is unpatentable under 35 U.S.C. § 103 if the claimed subject matter as a whole would have been obvious to a person of ordinary skill in the art at the time of the alleged invention. I also understand that an obviousness analysis takes into account the following factors, which are sometimes referred to as the *Graham* factors: (1) the scope and content of the prior art, (2) the differences between the claimed subject matter and the prior art, (3) the level of

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

ordinary skill in the art at the time of the invention, and (4) “objective indicia of non-obviousness,” also referred to as secondary considerations of non-obviousness. Those objective indicia include considerations such as whether a product covered by the claims is commercially successful due to the merits of the claimed invention, whether there was a long felt need for the solution provided by the claimed invention, whether others failed to find the solution provided by the claimed invention, whether others copied the claimed invention, and whether there was acceptance by others of the claimed invention as shown by praise from others in the field.

26. In determining the scope and content of the prior art, it is my understanding that a reference is considered appropriate prior art if it falls within the field of the inventor’s endeavor. In addition, a reference is appropriate prior art if it is reasonably pertinent to the particular problem with which the inventor was involved. A reference is reasonably pertinent if it logically would have commended itself to an inventor’s attention in considering his problem. If a reference relates to the same problem as the claimed invention, that supports use of the reference as prior art in an obviousness analysis.

27. To assess the differences between prior art and the claimed subject matter, it is my understanding that 35 U.S.C. § 103 requires the claimed invention to be considered as a whole. This “as a whole” assessment requires showing that one of ordinary skill in the art at the time of invention, confronted by the same

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

problems as the inventor and with no knowledge of the claimed invention, would have selected the elements from the prior art and combined them in the claimed manner.

28. It is my further understanding that the Supreme Court has recognized several rationales for combining references or modifying a reference to show obviousness of claimed subject matter. Some of these rationales include: combining prior art elements according to known methods to yield predictable results; simple substitution of one known element for another to obtain predictable results; a predictable use of prior art elements according to their established functions; applying a known technique to a known device (method or product) ready for improvement to yield predictable results; choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success; and some teaching, suggestion, or motivation that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention.

29. I understand that an assessment of what a reference discloses or teaches—for purposes of an anticipation analysis or an obviousness analysis—must be conducted from the perspective of a POSITA at the time of the invention. In other words, a reference discloses or teaches a claim limitation if a POSITA would, at the relevant time, interpret the reference as expressly, implicitly, or inherently

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

disclosing the claim limitation. I further understand that a reference does not need to use the exact language of the claim to disclose a claim limitation.

30. I understand that a patent owner may attempt to establish a date of invention that is earlier than the effective filing date of the patent. However, I understand that ACS is not attempting to establish an earlier invention date for the purpose of this IPR, and, thus, for the purpose of this IPR, the time of the invention (also called the “relevant time” herein) is the effective filing date of the ’768 patent. I understand that the earliest possible effective filing date of the ’768 patent is June 13, 2006 and that Petitioner has not alleged that the ’768 patent is not entitled to the June 13, 2006 filing date. Accordingly, I have assumed that June 13, 2006 is the time of invention or relevant time for purposes of this IPR.

SUBSTANTIVE TESTIMONY

A. The field of the invention and the level of ordinary skill in the art.

31. I was asked by counsel to assess the field of the invention of the ’768 patent and the level of ordinary skill in the art. In making this assessment, I have considered the ’768 Patent, my experience, and the testimony of Petitioner’s expert, Dr. Tufo.

32. In my opinion, the field of the invention of the ’768 patent is “cluster computing.” This is supported by the “Field of Disclosure” section of the patent, the specification generally, and the claims. The “Field of Disclosure” section states:

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

“The present disclosure relates to the field of cluster computer generally” The specification of the ’768 patent consistently discloses aspects of a computer cluster or methods of operating a computer cluster. And every challenged claim expressly recites a “computer cluster.”

33. Dr. Tufo did not expressly specify the field of the invention of the ’768 patent. However, Dr. Tufo refers generally to “parallel and distributed computing.” Ex. 1005 ¶ 40; *see also id.* ¶¶ 22-36. In my view, both “parallel computing” and “distributed computing” are different from the “cluster computing” field identified and claimed by the ’768 patent. A POSITA would understand that “parallel computing” is a category that includes “cluster computing” but “distributed computing” is different from both “cluster computing” and “parallel computing.”

34. Accordingly, I disagree with Dr. Tufo’s opinion that a POSITA “would have had a Bachelor’s degree in computer science, electrical engineering, or an equivalent field, and two years of academic or industry experience in *parallel and/or distributed computing*.” Ex. 1005 ¶ 40 (emphasis added). In my opinion, a POSITA would have had a Bachelor’s degree in computer science, electrical engineering, or an equivalent field, and two years of academic or industry experience in *cluster computing*. In conducting my analysis set forth in this Declaration, I applied my identification of the level of ordinary skill in the art, rather than Dr. Tufo’s.

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

However, my conclusions would not change even assuming that Dr. Tufo's identification of the level of ordinary skill in the art were correct.

35. At the time of the invention and presently I did and do have more training, expertise, and knowledge than a POSITA. However, I understand what a POSITA would have known and understood at the relevant time and my testimony herein is from the perspective of a POSITA at the relevant time.

B. Fundamental features of the claimed invention.

36. The '768 patent describes and claims a fundamental improvement over previous efforts to allow multiple nodes to work together to perform computations in parallel. Specifically, the inventors designed a mechanism to allow the nodes of a computer cluster to communicate tasks and data with one another in a peer-to-peer architecture. According to the background section of the patent, one previous attempt to allow multiple nodes to work together to perform computations was "a form of grid computing known as 'distributed computing.'" Ex. 1001 at 1:44-62. The form of grid computing described in the patent relied on a "master node that manages a plurality of slave nodes or computational nodes." *Id.* at 1:51-53. The "master kernel . . . handles all input, output, and scheduling of the other kernels (the computational kernels or slave kernels). Computational kernels receive commands and data only from the node running the master kernel." *Id.* at 1:55-59. The nodes "generally do not communicate with one another as peers." *Id.* at 1:46-47. The

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

patent distinguishes this form of grid computing from a “computer cluster having peer-to-peer node architecture.” *Id.* at 12:33-40.

37. The ’768 patent expressly claims the “peer-to-peer node architecture” disclosed in the specification. Specifically, claim 1 recites a computer cluster with multiple nodes and “a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture.” Claim 1 further recites that “one or more of the nodes are configured to . . . after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other.”

38. The specification discloses a particular mechanism for the nodes to communicate using a “peer-to-peer architecture.” Specifically, multiple “cluster node modules” establish connections from each cluster node module to every other cluster node module and exchange messages in a process that “provides the peer-to-peer behavior of the cluster node modules.” Ex. 1001 at 23:51-52, 24:39-40, 24:52-53, 25:1-2, 25:9-10, 25:23-24, 25:37-38. Dependent claim 4 recites the “cluster node modules.”

39. A POSITA would understand, in view of the specification, that “peer-to-peer behavior” is an essential feature of both the “peer-to-peer architecture” limitation of claim 1 and the narrower “cluster node modules” limitation of claim 4. The “peer-to-peer behavior” includes at least that each node can communicate tasks

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

and data with other nodes without the tasks and data being required to go through a central server or master node.

C. Claim construction of “a mechanism for the nodes to communicate . . . with each other using a peer-to-peer architecture” (all challenged claims)

40. Claim 1 recites “*a mechanism for the nodes to communicate* results of mathematical expression evaluation *with each other using a peer-to-peer architecture*” and “wherein one or more of the nodes are configured to . . . communicate at least some of the user instructions using *the mechanism for the nodes to communicate with each other*” (emphases added). With respect to these two limitations, there are two claim construction issues relevant to the Board’s decision whether to institute an IPR: (1) determining that both limitations refer to the same “mechanism for the nodes to communicate . . . with each other using a peer-to-peer architecture” and (2) determining the meaning of “peer-to-peer architecture.”

1. The “wherein” clause refers back to the same “mechanism for the nodes to communicate . . . with each other using a peer-to-peer architecture” introduced earlier in the claim.

41. A POSITA would understand that the “wherein” clause uses the standard claim-drafting technique of using a shorthand phrase introduced by the definite article “the”—specifically, “the mechanism for the nodes to communicate with each other”—to refer to a limitation recited earlier in the claim. A POSITA

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

would also understand that the **only** earlier limitation of claim 1 that recites a “mechanism” that provides antecedent basis for “the mechanism” of the “wherein” clause, is “a mechanism for the nodes to communicate . . . with each other using a peer-to-peer architecture.” Accordingly, in my view, the “wherein” clause refers back to the same “mechanism for the nodes to communicate . . . with each other using a peer-to-peer architecture” recited earlier in the claim. This means that “results of mathematical expression evaluation” and “at least some of the user instructions” must be communicated using the claimed “mechanism for the nodes to communicate . . . with each other using a peer-to-peer architecture.”

2. “peer-to-peer architecture”

42. The claimed communication mechanism must use a “peer-to-peer architecture.” A POSITA would understand that, in the context of cluster computing, the adjective “peer-to-peer” ordinarily means that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node. A POSITA would also look to the specification of the ’768 patent to confirm this understanding.

43. As explained below, a POSITA would conclude from the specification that communicating tasks and data with other nodes without the tasks and data being required to go through a central server or master node is a primary function of the “peer-to-peer architecture” of the ’768 patent. Therefore, “peer-to-peer

IPR2021-00019

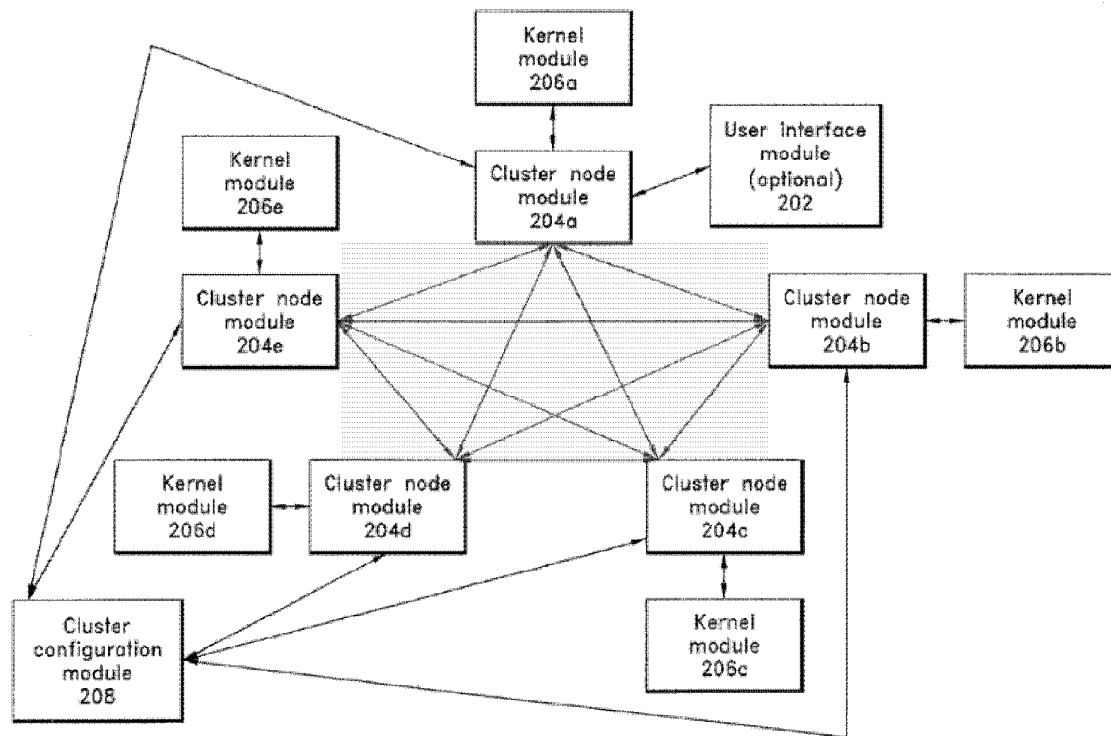
NVIDIA Corp. v. Advanced Cluster Sys.

architecture,” in the context of the ’768 patent, should be construed to require at least “an architecture in which each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.”

44. A POSITA would understand that the “peer-to-peer architecture” required by claim 1 is one feature of the interconnected “cluster node modules” disclosed by the ’768 patent and recited by claim 4.¹ Accordingly, the ’768 patent’s disclosure of the “peer-to-peer architecture” enabled by the “cluster node modules” is instructive of what “peer-to-peer architecture” means in the context of the ’768 patent. Figure 2 illustrates that each cluster node module is connected to every other cluster node module, as shown by the yellow highlighting in the annotated figure below.

¹ Claim 1 is broader than claim 4 because, while it requires a “peer-to-peer architecture,” it does not require that architecture to be implemented using the “cluster node modules” recited by claim 4.

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.*FIG. 2*

Ex. 1001, Fig. 2 (highlighting added). The specification confirms that “each cluster node module 204a-e is connected to all other cluster node modules” and “[t]he cluster node modules 204a-e establish communication with one another” through “direct connections” between each cluster node module. Ex. 1001 at 23:13-14, 23:51-56.

45. As illustrated and described, the cluster node modules provide a direct connection from each node to every other node. In addition, the specification discloses that a process for passing messages among the cluster node modules “provides the peer-to-peer behavior of the cluster node modules,” allowing them to “interact on a pair-wise or collective basis.” Ex. 1001 at 25:22-41.

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

46. In view of the specification, a POSITA would understand that the message-passing process allows each cluster node module to communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node. Specifically, the specification discloses that “[c]ommunications can occur between any two or more cluster node modules” and that “[e]ach of the cluster node modules 204a-e is in communication with respective kernel modules 206a-e,” thereby enabling “MPI calls and advanced cluster commands” to be “used to parallelize program code received from an optional user interface module 208 and *distribute tasks* among the kernel modules 206a-e.” Ex. 1001 at 6:9-25 (emphasis added).

47. In addition, the specification distinguishes a peer-to-peer architecture from the master-slave architecture typically used for “a form of grid computing known as ‘distributed computing.’” Ex. 1001 at 1:44-62; *see also id.* at 12:30-33 (gridMathematica connects kernels “in a master-slave relationship rather than a peer-to-peer relationship”). The specification describes the following characteristics of the master-slave architecture of grid computing:

Grid computers include at least one node known as a master node that manages a plurality of slave nodes or computational nodes. In gridMathematica, each of a plurality of kernels runs on a single node. *One kernel is designated the master kernel, which handles all input, output, and scheduling of the other kernels (the computational*

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

kernels or slave kernels). Computational kernels receive commands and data only from the node running the master kernel.

Id. at 1:51-59 (emphasis added). By distinguishing the peer-to-peer architecture of the invention from the master-slave architecture, the specification implicitly indicates that the peer-to-peer architecture does not have the identified characteristics of the master-slave architecture. Accordingly, by contrast to the master-slave architecture, each node in the disclosed peer-to-peer architecture is able to handle “scheduling” and communicating (including distributing, sending, and receiving) “commands and data” to and from other nodes without requiring them to go through a central server or master node. A POSITA would understand that the communication of tasks and data falls within these “scheduling” and communicating “commands and data” functions.

48. Accordingly, a POSITA would understand that the communication of tasks and data is a primary function that the nodes of the disclosed peer-to-peer architecture of the ’768 patent can handle without requiring a central server or master node. The Board should construe “peer-to-peer architecture,” in the context of the ’768 patent, to require at least “an architecture in which each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.”

IPR2021-00019

*NVIDIA Corp. v. Advanced Cluster Sys.***D. “cluster node module” (claim 4)**

49. Dependent claim 4 recites “wherein each of the nodes comprises one or more cluster node modules.” The phrase “cluster node module” was not a common or well-known technical phrase having any specific meaning in the relevant art at the time of the invention. Indeed, even today, more than 14 years after the earliest effective filing date of the ’768 patent, a search for patents or patent application publications that use the phrase “cluster node module” returns only patents and applications filed by the inventors of the ’768 patent. Accordingly, there was no ordinary meaning of “cluster node module” in the relevant field at the relevant time. Further, while the individual terms “cluster,” “node,” and “module,” were known in the relevant field, a POSITA would understand that no ordinary meaning for the combined phrase “cluster node module” could reliably be composed by attempting to combine the meanings of the individual terms.

50. In view of the claim language and specification, a POSITA would understand that the inventors coined the phrase “cluster node module” to encapsulate essential features of the modules that interconnect the nodes in a preferred embodiment of the invention.² I understand that, because “cluster node module” is

² Several claims of the ’768 patent, including claim 1, recite “a plurality of nodes” but not “cluster node modules.”

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

a coined phrase, reliance on the specification is necessary to ascertain its meaning. As explained below, in view of the specification, the Board should construe “cluster node module” to mean “a module that cooperates with other cluster node modules to establish intercommunication among nodes in a computer cluster and to exchange messages such that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.”

51. The specification discloses several *optional* components of the cluster node modules. For example:

FIG. 3 shows one embodiment of a cluster node module 204 implementing MPI calls and advanced MPI functions. ***In the embodiment shown in FIG. 3***, cluster node module 204 includes MPI module 302, advanced functions module 304, received message queue 306, and message receiving queue 308.

Ex. 1001 at 12:41-46 (emphases added). Because the components depicted in Figure 3 are part of “one embodiment,” a POSITA would understand that they are optional components of the cluster node modules. Dependent claims of U.S. Patent No. 8,082,289 (which is related to the ’768 patent) specifically reciting that the cluster node modules comprise an MPI module, advanced functions module, received message queue, and message receiving queue also demonstrate that these components are optional parts of the cluster node modules. IPR2020-01608, Ex.

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

1001, claim 8 (received message queue); claim 9 (message receiving queue); claim 13 (advanced functions module); claim 26 (MPI module). Accordingly, a POSITA would understand that the basic “cluster node modules” do not require an MPI module, advanced functions module, received message queue, and message receiving queue. Thus, if the Petition suggests that a “cluster node module” necessarily comprises these components (*see* Pet. at 52), the Petition is incorrect.³

52. Rather than interpreting “cluster node module” to include optional components such as those depicted by Figure 3, a POSITA would look to the specification to limit the “cluster node module” to its essential features. The specification unambiguously discloses that cluster node modules are configured to establish intercommunication with one another, communicate messages among themselves and with kernels, and, through such message-passing, “provide[] the peer-to-peer behavior of the cluster node modules.” Ex. 1001 at 23:51-52, 24:39-40, 24:52-53, 25:1-2, 25:9-10, 25:23-24, 25:37-38. Significantly, the specification does not say these disclosed features apply just to “one embodiment” of the cluster node

³ The Petition does not commit to *any* definitive claim construction of “cluster node module.” The Petition merely states that “a ‘cluster node module’ includes code *relating to* . . . and *can* also include” the components depicted by Figure 3. Pet. at 52.

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

modules or otherwise suggest they are optional. Therefore, a POSITA would understand that the capability to establish intercommunication among all cluster node modules and to exchange messages to provide “peer-to-peer behavior” are essential features of the cluster node modules.

53. A POSITA would further examine both the ordinary meaning of “peer-to-peer” and the specification to determine what is meant by the cluster node modules’ “peer-to-peer behavior.” As explained above with respect to the “peer-to-peer architecture” limitation, a POSITA would conclude that “peer-to-peer behavior” includes at least that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.

54. Accordingly, the Board should construe “cluster node module” to mean “a module that cooperates with other cluster node modules to establish intercommunication among nodes in a computer cluster and to exchange messages such that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.”⁴

⁴ A POSITA would recognize there is some overlap between the “cluster node modules” limitation of claim 4 and the “peer-to-peer architecture” limitation of claim 1, but the “cluster node modules” limitation is narrower.

IPR2021-00019

*NVIDIA Corp. v. Advanced Cluster Sys.***E. The Petition does not show that the prior art discloses the “peer-to-peer architecture” limitation (all challenged claims).**

55. Claim 1 recites “a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using a *peer-to-peer architecture*” (emphasis added). As explained above, the Board should construe “peer-to-peer architecture” to require at least “an architecture in which each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.” The alleged prior art does not disclose the “peer-to-peer architecture” limitation at least because it expressly requires at least all *tasks* to go through a central server or master node.

56. Specifically, Schreiner 1 discloses: “All remote schedulers send new tasks to the root node scheduler which distributes them among all machines.” Ex. 1008 at 316. It further discloses that “[s]ince the root is in charge of task scheduling, the root sees every task created in the session.” *Id.* at 323. A POSITA would understand that the “root node” of Schreiner 1 is a “master node.” Accordingly, the prior art’s requiring at least tasks to go through a master node means that the prior art does not disclose a “peer-to-peer architecture.”

57. The Petition relies on Schreiner 1’s disclosure that a node needing “to send a message to one of its peers . . . can thus establish a direct connection for message transfers” and Schreiner 3’s nearly identical disclosure that “[w]hen a client

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

wants to send a message to another client through a peer connection, but such a connection has not existed between these two clients yet, the client creates it.” Pet. at 29-30. But these disclosures do not negate Schreiner 1’s express disclosure that its architecture requires all tasks to go through a master node. A POSITA would understand that the creation of a “peer connection” does not make Schreiner’s architecture “peer-to-peer” when Schreiner 1 and Schreiner 3 expressly disallow communication of tasks using a peer-to-peer architecture by requiring all tasks to go through a master node.

58. Nothing in the prior art even suggests that tasks can be communicated using direct node-to-node communication without requiring the tasks to go through the “single root node.” Indeed, Schreiner 1’s express disclosure that “[a]ll remote schedulers send new tasks to the root node scheduler which distributes them among all machines” (Ex. 1008 at 316) and “the root is in charge of task scheduling” (*id.* at 323) undermines any possible contention that the disclosed architecture enables direct node-to-node communication of tasks the tasks being required to go through the “single root node.”

59. Petitioner’s arguments about the communication of result messages fail to show that the prior art satisfies the “peer-to-peer architecture” limitation at least because they do not negate the prior art’s express teaching that tasks are required to go through a master node. Further, the Petition even fails to show that the prior art

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

communicates result messages without requiring the involvement of a master node.

Petitioner attempts to imply there is a link between Schreiner 1's sentence, on page 319, about an "idle kernel" returning "a message result" with the sentence on page 311 that "additional socket connections between remote scheduler instances are created on demand" by quoting those sentences directly adjacent to each other in the Petition. Pet. at 30. But those sentences are neither adjacent nor logically linked in Schreiner 1. The sentences are eight pages apart, in different sections, and Schreiner 1 does not suggest they are related to each other. The page 311 quote is in the "Software architecture" section and relates to the "software architecture" of Figure 1. Ex. 1008 at 311. The page 319 quote, by contrast, is in the "System and execution model" subsection of the "Fault tolerance" section and relates to the "execution model" of Figure 5. *Id.* at 319. Significantly, the "execution model depicted in Fig. 5," as shown below, does ***not*** depict any direct node-to-node communications that do not require data to go through the root node:

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

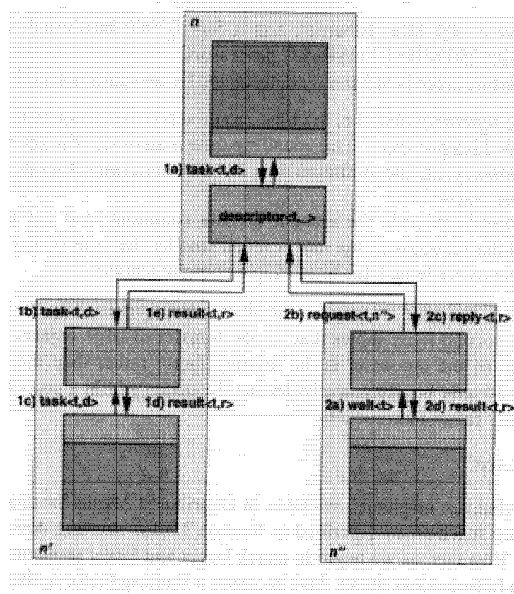


Fig. 5. Execution model.

Id. at 320. Indeed, *every* connection depicted on Figure 5 includes the root node.

60. The Petition also relies on Figure 1 of Schreiner 3, arguing that the “6 result message” box of depicts “client n” sending “client 1” a “result message.” Pet. at 31. However, the Petition ignores the “7 store message” box, which shows that “client n” sends the same “result” to the “root.” Ex. 1010 at 5 (“When the result is computed the root receives it in a *store message* and saves it to the hard disk (see Figure 1).”); 9 (“If a node has computed the result of a task, it sends it in a *result message* to the node which created this task and also sends a duplicate in a *store message* to the root.”). Accordingly, Schreiner 3 sends results to a master node. Further, Schreiner 3’s communication of results does not overcome the prior art’s express disclosure that tasks are required to go through a master node. Therefore, the Petition does not prove that the prior art discloses a “peer-to-peer architecture”

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

in which “each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.”

61. I understand that the Petition does not allege that a POSITA would modify Schreiner 1 or any of the other Distributed Maple references to not require the root node to be involved in the distribution of tasks. Accordingly, I understand it is not necessary to analyze whether it would have been obvious to make such a modification to assess the Petition’s arguments. Nevertheless, I further analyzed the Distributed Maple references and concluded that a POSITA would understand that Schreiner 1’s disclosure that tasks must be distributed through the root node is a fundamental feature of Distributed Maple that a POSITA would not have modified, at least because that feature enables the fault tolerance mechanism disclosed in Schreiner 1. As part of its fault tolerance mechanism, Schreiner 1 discloses a logging mechanism in which “the root sees every result computed in the session” and that “the root sees every task created in the session.” Ex. 1008 at 323. Schreiner 1 further discloses that “the delegation of all logging activities to a single root node that also performs the task scheduling decisions” are important features that enable the fault tolerance mechanism to be “simple” and to avoid “global snapshots as required in message passing programs” and that this logging mechanism has “runtime overhead” that is “very moderate” and the fault tolerance mechanism does not require much extra overhead.” A POSITA would understand, in view of the

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

disclosure of Schreiner 1, that distributing tasks through the root node is required to take advantage of the disclosed fault tolerance mechanism. Therefore, even if fault tolerance were optional at the user level—that is, some users could turn on the provision of fault tolerance while others could turn it off—a POSITA would not modify Distributed Maple to not require tasks to go through the root node, because such a modification would destroy the ability of users to use the fault tolerance mechanism of Distributed Maple.

F. The Petition does not show that the prior art discloses communication of “user instructions” using the “peer-to-peer architecture” (all challenged claims).

62. As explained above, the “wherein” clause of claim 1 requires the same “peer-to-peer architecture” used to communicate “results” to also communicate “at least some of the user instructions.” The Petition does not show that the prior art meets this limitation. The Petition relies on the prior art’s communication of *tasks* as allegedly meeting the limitation, but as explained above, the prior art expressly discloses that its communication of tasks relies on requiring the tasks to go through a root node, meaning that it does not use a peer-to-peer architecture.

63. The Petition specifically alleges that, in the prior art, “the root node accepts . . . Distributed Maple instructions” and then the “root node communicates these instructions using the node-to-node messaging mechanism.” Pet. at 49-50 (citing Ex. 1008 (Schreiner 1) at 309-314). But Schreiner 1 never says that those

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

instructions are communicated using a “peer-to-peer architecture” in which the instructions are not required to go through a central server or master node. Ex. 1008 at 309-314.

64. In fact, the cited pages of Schreiner 1 indicate that instructions are sent as *tasks*. *Id.* For example, the Petition acknowledges that “the user’s instruction dist[start] is communicated to other nodes as *task* messages.” Pet. at 50 (emphasis added). The Petition further asserts that dist.maple source code confirms that these instructions are communicated as *task* messages. *Id.* (emphasis added). However, as explained above, Schreiner 1 expressly discloses that tasks, even if viewed as task messages, are distributed by the root node: “All remote schedulers send new tasks to the root node scheduler which distributes them among all machines.” Ex. 1008 at 316; *see also id.* at 323 (“the root is in charge of task scheduling”). Accordingly, because the communication of all tasks must go through a master node, the prior art does not disclose using a “peer-to-peer architecture” for such communication.

65. I understand that the Petition does not allege that a POSITA would modify Schreiner 1 or any of the other Distributed Maple references to not require the root node to be involved in the distribution of tasks. Accordingly, I understand it is not necessary to analyze whether it would have been obvious to make such a modification to assess the Petition’s arguments. Nevertheless, I further analyzed the Distributed Maple references and concluded that a POSITA would understand that

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

Schreiner 1's disclosure that tasks must be distributed through the root node is a fundamental feature of Distributed Maple that a POSITA would not have modified, at least because that feature enables the fault tolerance mechanism disclosed in Schreiner 1. As part of its fault tolerance mechanism, Schreiner 1 discloses a logging mechanism in which "the root sees every result computed in the session" and that "the root sees every task created in the session." Ex. 1008 at 323. Schreiner 1 further discloses that "the delegation of all logging activities to a single root node that also performs the task scheduling decisions" are important features that enable the fault tolerance mechanism to be "simple," to avoid "global snapshots as required in message passing programs" and to have "runtime overhead" that is "very moderate."

Id. A POSITA would understand, in view of the disclosure of Schreiner 1, that distributing tasks through the root node is required to take advantage of the disclosed fault tolerance mechanism. Therefore, even if fault tolerance were optional at the user level—that is, some users could turn on the provision of fault tolerance while others could turn it off—a POSITA would not modify Distributed Maple to not require distributing tasks through the root node, because such a modification would destroy the ability of users to use the fault tolerance mechanism of Distributed Maple.

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

G. The Petition does not show that the prior art discloses “wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node” (all challenged claims).

66. This limitation recites a precise order of operations involving three specific nodes: (1) the third node receives, from the second node, a result of a calculation that was performed by the second node in a different claim limitation; (2) the third node performs a different calculation based on the received result; and (3) the third node sends the new result to the first node. The Petition alleges that the alleged prior art publications “disclose this element.” Pet. at 43.

67. The Petition fails to prove its allegation that the prior art discloses this detailed limitation. To prove that allegation, the Petition would need to show where the prior art teaches the precise order of operations, involving three specific nodes, specified by the claim. The Petition does not do that. Instead, the Petition refers to the prior art’s general disclosure that multiple nodes may be involved in evaluating mathematical expressions and then speculates that the prior art could work in a manner that would meet the claim limitation. Pet. at 44-47.

68. The Petition first relies on “the example of Figure 5” of Schreiner 1. *Id.* at 45. It is clear on the face of Petitioner’s argument that Figure 5 does not

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

actually disclose every detail about the claim limitation and that Petitioner is merely speculating about how Figure 5 allegedly could work. The Petition asserts that a “POSITA would understand that the third node then *could* execute a second mathematical expression using that result, and in fact, that is *likely why* the third node requested the result in the first place.” *Id.* at 46 (emphases added). The Petition further speculates that “the third node *could* send the result of the second mathematical expression to the first (root) node.” *Id.* (emphasis added).

69. The Petition also relies on an “example . . . illustrated in Schreiner 3 Fig. 1.” *Id.* at 47. But the cited example from Schreiner 3 is about a mechanism for “the *logging* of task return values” (Ex. 1010 at 1) and neither Schreiner 3 nor the Petition explains how Figure 1 relates to the evaluation of mathematical expressions by multiple nodes. Accordingly, Petitioner’s interpretation of Figure 1 is at best speculative. Moreover, the Petition itself makes clear that Figure 1 does not actually disclose every detail about the claim limitation and that Petitioner is merely speculating about how Figure 1 allegedly could work. The Petition speculates that “once [client 1] receives the result message from node ‘client n,’ it *may* use it to perform at least a second mathematical expression evaluation.” Pet. at 47. Petitioner offers no evidence or explanation supporting this speculation. *Id.*

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

70. Because the Petition merely speculates about what prior art nodes “could” or “may” do, the Petition fails to prove Petitioner’s assertion that the prior art discloses the claim limitation.

H. The Petition does not show that the prior art discloses “cluster node modules” (claims 4, 6-10, 30, and 31).

71. Claims 4, 6-10, 30, and 31 indirectly depend on claim 1 and, thus, are patentable for the same reasons set forth above that claim 1 is patentable. In addition, these claims are patentable because the prior art does not disclose the limitation of claim 4 reciting that each node “comprises one or more cluster node modules.” As explained above, because “cluster node module” is a coined phrase encapsulating the essential features of the cluster node modules disclosed in the specification, the phrase should be construed to mean “a module that cooperates with other cluster node modules to establish intercommunication among nodes in a computer cluster and to exchange messages such that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.” These essential features of the cluster node modules are fundamental parts of the claimed invention of claim 4.

72. The Petition alleges that the “dist.Scheduler and dist.maple components” of the prior art are “cluster node modules.” Pet. at 52-55. However, the Petition does not allege or prove that those components allow each node to

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

“communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.” *Id.* In fact, Schreiner 1 expressly discloses that at least distributing tasks requires a master node: “All remote schedulers send new tasks to the root node scheduler which distributes them among all machines.” Ex. 1008 at 316. Schreiner 1 further confirms “the root is in charge of task scheduling” and “the root sees every task created in the session.” *Id.* at 323. A POSITA would understand that the “root node” of Schreiner 1 is a “master node.” Accordingly, because the purported cluster node modules of the prior art do not “communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node,” they do not satisfy the “cluster node module” limitation as properly construed.

73. I understand that the Petition does not allege that a POSITA would modify Schreiner 1 or any of the other Distributed Maple references to not require the root node to be involved in the distribution of tasks. Accordingly, I understand it is not necessary to analyze whether it would have been obvious to make such a modification to assess the Petition’s arguments. Nevertheless, I further analyzed the Distributed Maple references and concluded that a POSITA would understand that Schreiner 1’s disclosure that tasks must be distributed through the root node is a fundamental feature of Distributed Maple that a POSITA would not have modified, at least because that feature enables the fault tolerance mechanism disclosed in

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

Schreiner 1. As part of its fault tolerance mechanism, Schreiner 1 discloses a logging mechanism in which “the root sees every result computed in the session” and that “the root sees every task created in the session.” Ex. 1008 at 323. Schreiner 1 further discloses that “the delegation of all logging activities to a single root node that also performs the task scheduling decisions” are important features that enable the fault tolerance mechanism to be “simple,” to avoid “global snapshots as required in message passing programs” and to have “runtime overhead” that is “very moderate.”

Id. A POSITA would understand, in view of the disclosure of Schreiner 1, that distributing tasks through the root node is required to take advantage of the disclosed fault tolerance mechanism. Therefore, even if fault tolerance were optional at the user level—that is, some users could turn on the provision of fault tolerance while others could turn it off—a POSITA would not modify Distributed Maple to not require distributing tasks through the root node, because such a modification would destroy the ability of users to use the fault tolerance mechanism of Distributed Maple.

H. Patent Owner’s SEMTM and SETTM products embody the challenged claims of the ’768 patent.

74. I have also been asked by Patent Owner to provide my opinion on whether Patent Owner’s SEMTM and SETTM products practice the challenged claims of the ’768 patent as they relate to objective indicia of non-obviousness. Based on

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

my review of technical documents which I understand were created near the time of the invention and accurately describe the SEMTM and SETTM products, and limited information about the SEMTM and SETTM products provided by Dr. Dean Dager, one of the primary developers of the SEMTM and SETTM products, it is my opinion that the SEMTM and SETTM products practice all limitations of at least claim 1 of the '768 patent.

75. In addition to the '768 patent, I reviewed the following documents in arriving at my conclusion that the SEMTM and SETTM products embody claim 1 of the '768 patent:

- SEMTM Poster (Ex. 2009);
- SEMTM PingPong Benchmark (Ex. 2010);
- SEMTM Manual (Ex. 2011);
- SEMTM White Paper (Ex. 2012);
- SETTM White Paper (Ex. 2013);
- SETTM Manual (Ex. 2014);
- SETTM Presentation (Ex. 2015);
- SETTM Datasheet (Ex. 2016).

76. Exhibit 2028 is a claim chart that accurately sets forth my analysis showing that the SEMTM product embodies at least claim 1 of the '768 patent. I adopt Exhibit 2028 as part of my testimony in this Declaration.

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

77. Exhibit 2029 is a claim chart that accurately sets forth my analysis showing that the SETTM product embodies at least claim 1 of the '768 patent. I adopt Exhibit 2029 as part of my testimony in this Declaration.

78. As identified in the claim charts, with respect to some aspects of certain claim limitations, I also considered limited information about the SEMTM and SETTM products provided by Dr. Dauger. In my view, the information provided by Dr. Dauger confirms that the SEMTM and SETTM products include architectural and functional features that are at least implied or suggested by the technical documents and that I would expect the SEMTM and SETTM products to have based on my review of the technical documents. Further, the technical documents are entirely consistent with, and fully support, the information provided by Dr. Dauger.

79. As shown in Exhibits 2028 and 2029, the SEMTM and SETTM products are software products that, when installed on a computer cluster, result in a computer cluster having the architectural and functional features included in claim 1, including the “peer-to-peer architecture” which enables each node to “communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node,” as required by the claim. As further explained in Exhibits 2028 and 2029, the SEMTM and SETTM products also practice the other limitations set forth in at least claim 1.

IPR2021-00019

NVIDIA Corp. v. Advanced Cluster Sys.

DECLARATION

80. I declare that all statements made herein on my own knowledge are true and that all statements made on information and belief are believed to be true, and further, that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Dated: 2/9/2021



Jaswinder Pal Singh, Ph.D